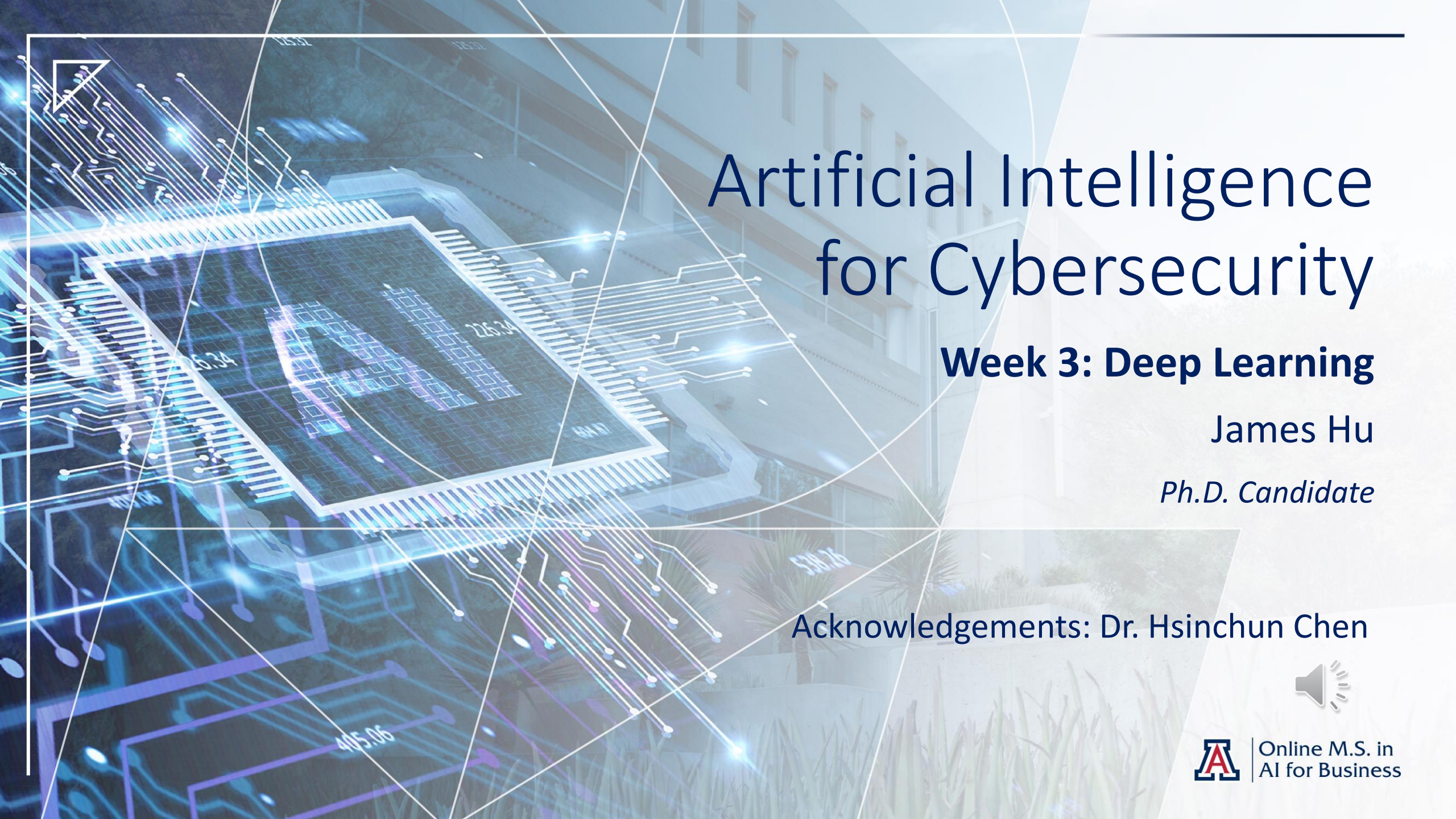




Artificial Intelligence for Cybersecurity

Week 3: Deep Learning

James Hu

Ph.D. Candidate

Acknowledgements: Dr. Hsinchun Chen



Online M.S. in
AI for Business

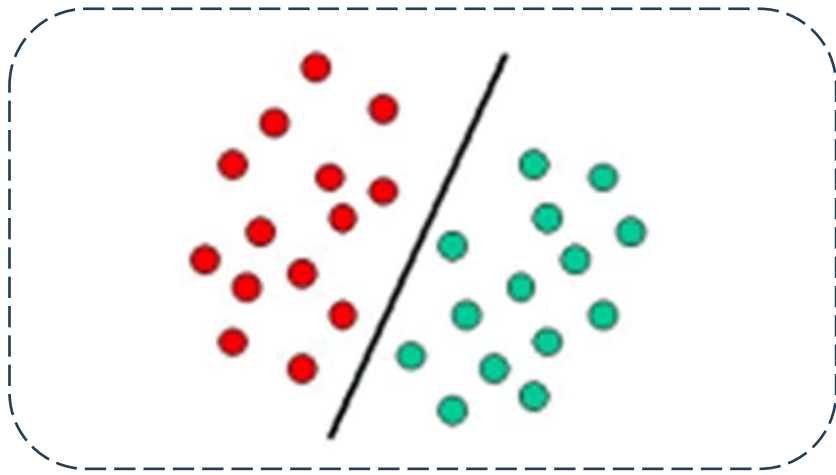


Today's Learning Objectives

- ▶ Understand the fundamental structure of deep learning models.
- ▶ Explain how artificial neural networks learn through the feedforward and backpropagation algorithms.
- ▶ Explain how MLP, CNN, and RNN handle different types of data.
- ▶ Understand the attention mechanism in Transformers.
- ▶ Explain how SFT and LoRA help task adaptation.



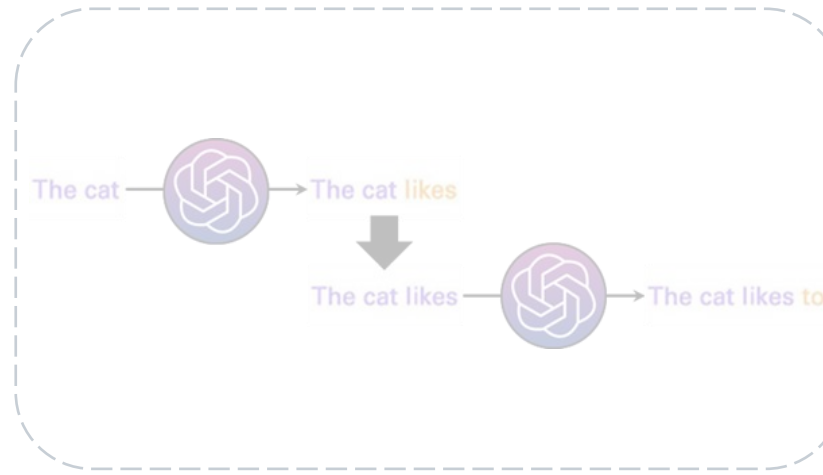
From Week 2: Predictive Analytics



Classification

Examples

- **Phishing Detection:** Flagging emails that have a high probability of being phishing
- **Malware Detection:** Determine which files are suspicious or malicious



Generation

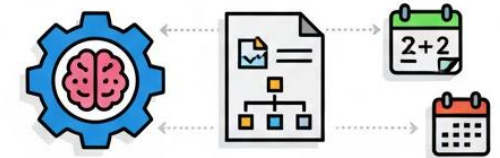
Examples

- **After Action Summarization:** Generate reports that summarize key points of a cybersecurity incident.

1. Data



2. Training & Testing



3. Prediction & Forecasting



Online M.S. in AI

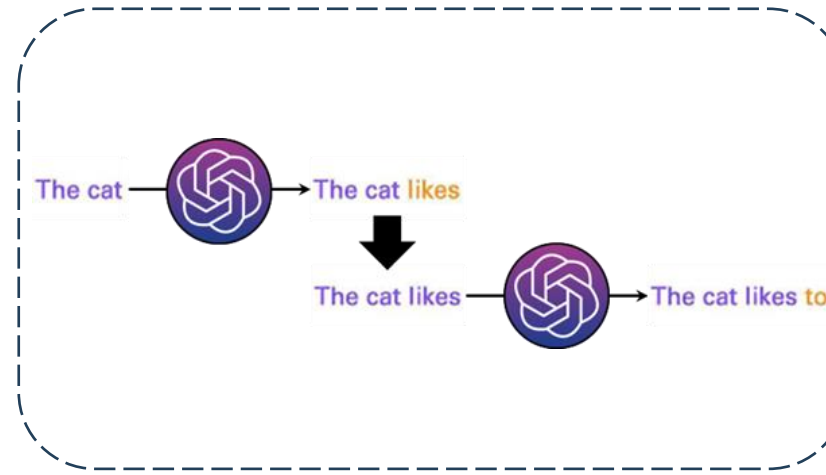
For Week 3: Predictive Analytics



Classification

Examples

- **Phishing Detection:** Flagging emails that have a high probability of being phishing
- **Malware Detection:** Determine which files are suspicious or malicious



Generation

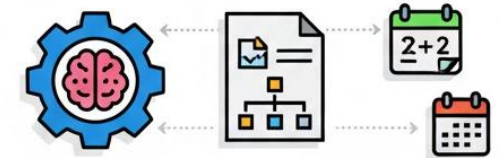
Examples

- **After Action Summarization:** Generate reports that summarize key points of a cybersecurity incident.

1. Data



2. Training & Testing



3. Prediction & Forecasting



Online M.S. in AI

From Week 2: Overview of ML/AI Models

- ▶ For the purposes of this class, we can describe an ML model (h) based on its:
 - ▶ **Input (x):** What the model perceives.
 - ▶ **Parameters (θ):** The model's "numerical dials" controlling how it answers.
 - ▶ **Output ($\hat{y} := h_{\theta}(x)$):** What happens inside the model to produce an answer.
 - ▶ **Loss Function ($\mathcal{L}(\hat{y})$):** How is the model's answer "wrong?"
 - ▶ **Optimization:** How the model changes its parameters to produce "less wrong" answers.
- ▶ **Core Intuition:** When learning, a model aims to find the optimal parameter (θ^*) that produces the overall least wrong answers.
 - ▶ Often happens iteratively, with the model slightly adjusting its parameters many times.



From Week 2: How do Machines Learn?

- ▶ How do humans learn?
- ▶ Imagine studying for a test. **You** may undertake the following steps:
 1. Review your **study notes** to **answer** a practice problem
 2. Grade the **answer**, focusing on your **mistakes**
 3. Revise your **understanding** based on your **mistakes**
 4. Repeat until you are ready for the test.
- ▶ Machine Learning (ML) is similar.
- ▶ To learn a task, **ML models** undergo training:
 1. **Model Input & Output:** Inputs a **datapoint** to produces an **output**.
 2. **Loss Calculation:** Use a loss function to calculate the **output's loss value**
 3. **Optimization:** Adjust the model's **parameters** based on the **loss value**.
 4. Repeat until the model converges.
- ▶ **AI often takes inspiration from humans and nature!**



Origins of Today's Favorite AI Model



Nature, 27 January 2016:

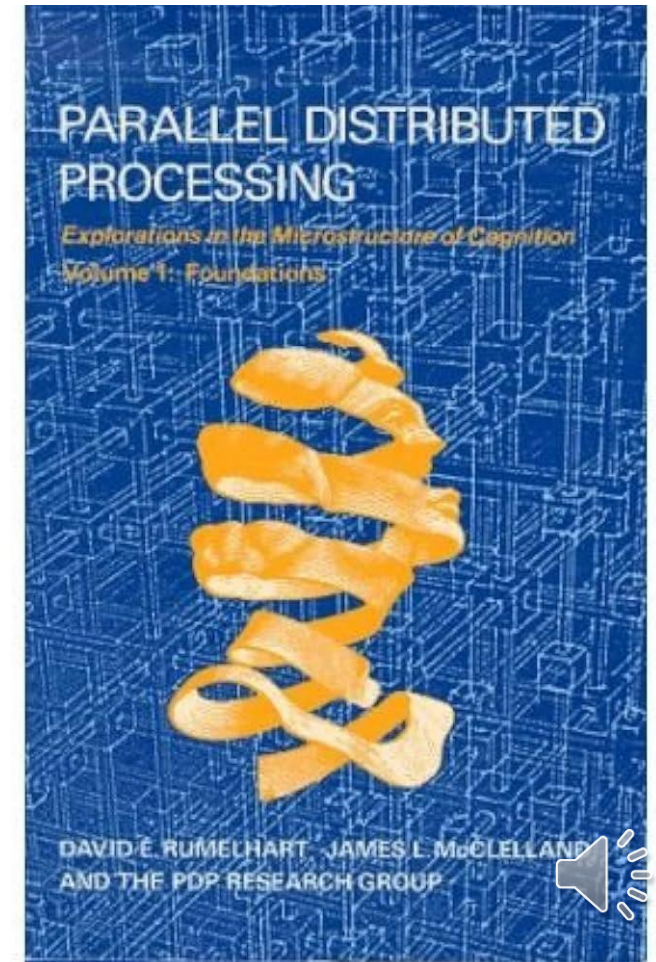
- ▶ “...AlphaGo program applied deep learning in **neural networks** (convolutional NN) — **brain-inspired** programs in which connections between layers of simulated neurons are strengthened through examples and experience.”



.S. in AI

Origins of Artificial Neural Networks

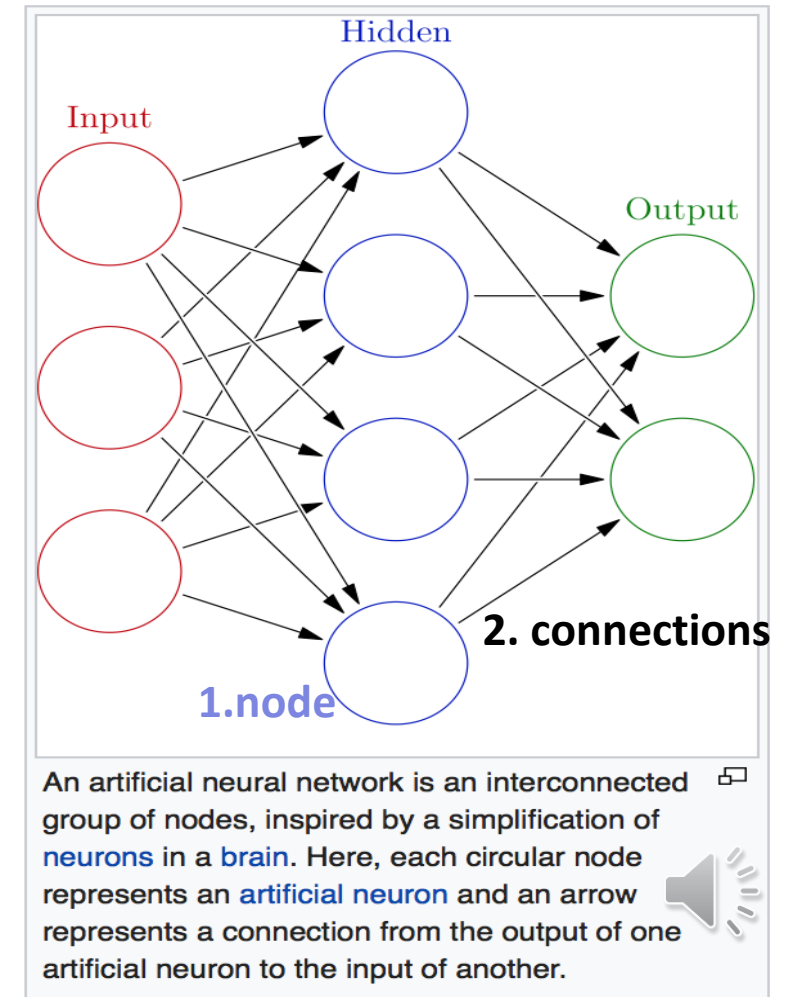
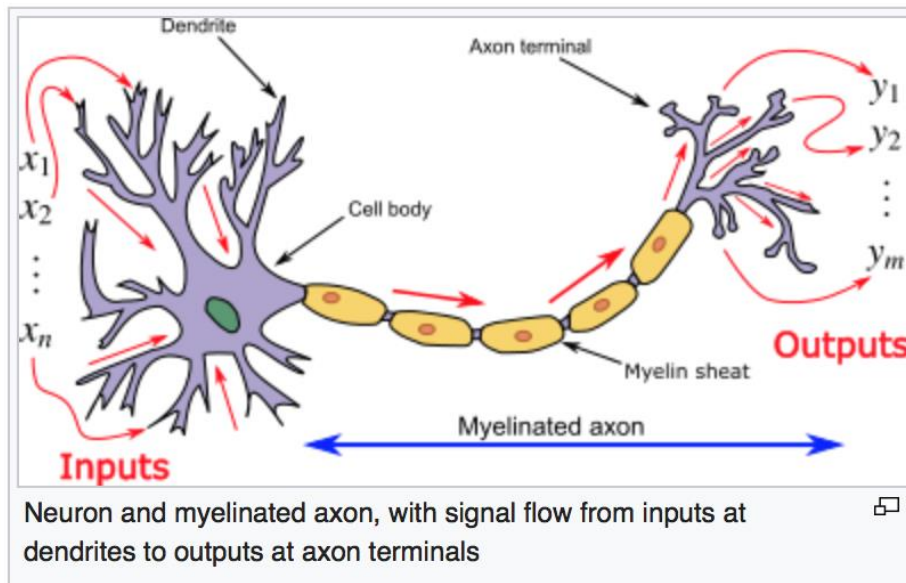
- ▶ Inspired by biological neural networks (animal brains), **Artificial Neural Networks (ANN)** was originated in 1950 to simulate intelligence through simple units and basic patterns.
- ▶ **John Hopfield** introduced **associative memory** in **1982**, using physics-based loops to store and retrieve information (become the foundation of recurrent neural networks).
- ▶ **PDP Research Group** at UCSD formalize the ANN concept in **1986** by modeling intelligence as distributed patterns across network connections (Rumelhart et al. 1986).
 - ▶ **Geoffrey Hinton** pioneered the **backpropagation** algorithm to train these patterns.
- ▶ **Hopfield and Hinton** were jointly awarded the **2024 Nobel Prize in Physics** for these foundational contributions to neural computation.



Artificial Neural Networks

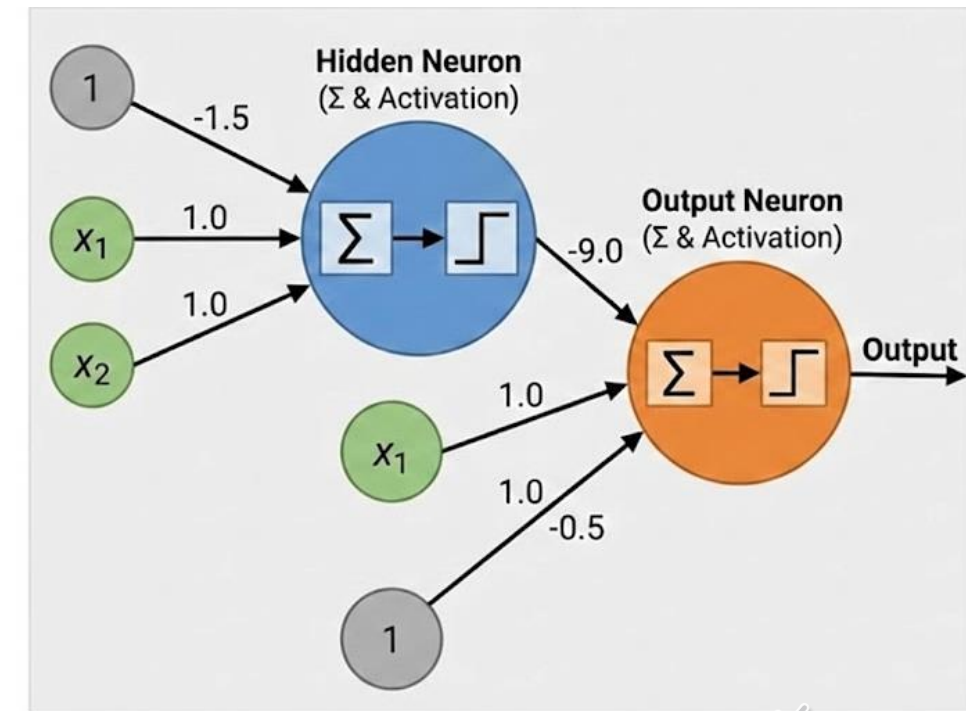
► Key Components in ANN:

1. **Nodes:** Artificial *neurons* (processing units)
2. **Connections:** Synapses (signal transmission)



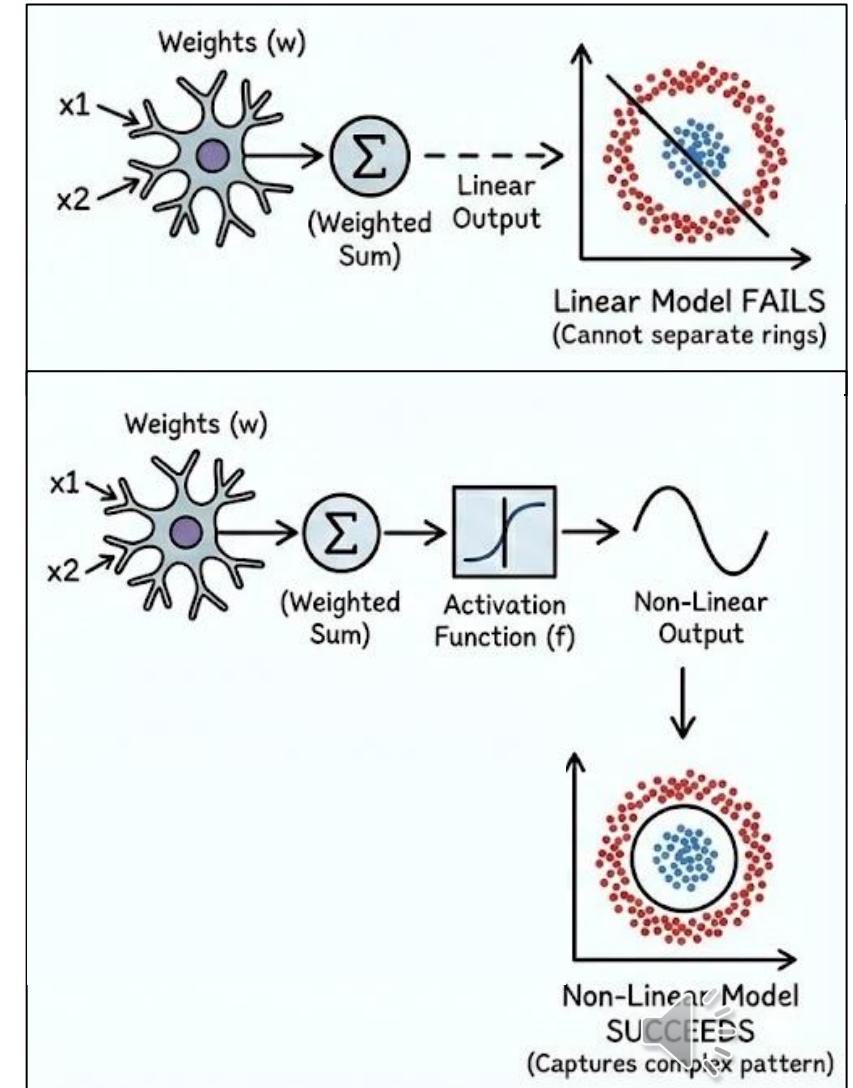
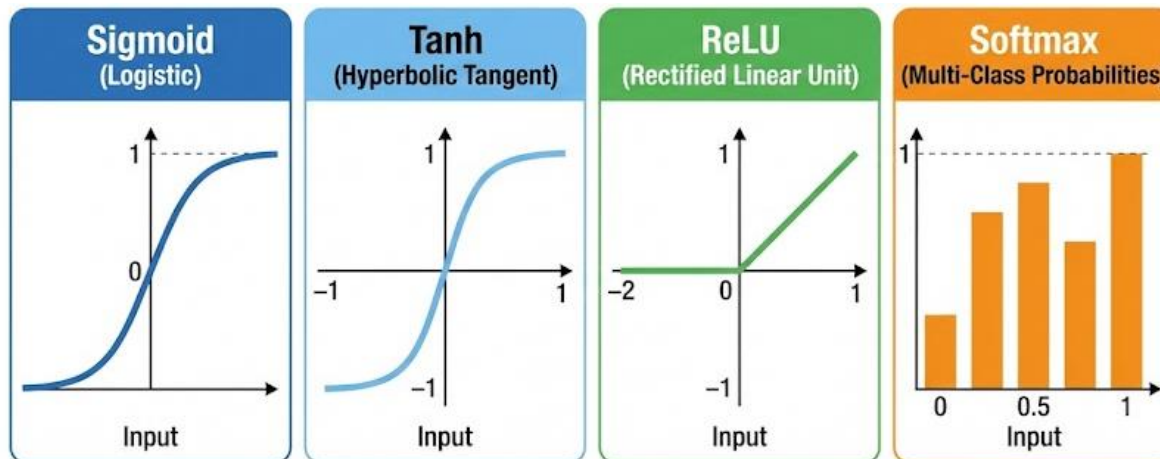
Artificial Neural Networks

- ▶ **A neuron** is a computational unit in the neural network that exchanges messages with other.
- ▶ Two components exist in each neuron:
 1. **Weighted sum:** Multiply each input by its weight and add them up
$$z = (1 \times -1.5) + (x_1 \times 1.0) + (x_2 \times 1.0)$$
 2. **Activation function $f(z)$:** Add non-linearity to the weighted sum.
- ▶ The result of these two components (neuron's output) is sent to the next neuron as its input through the connections.



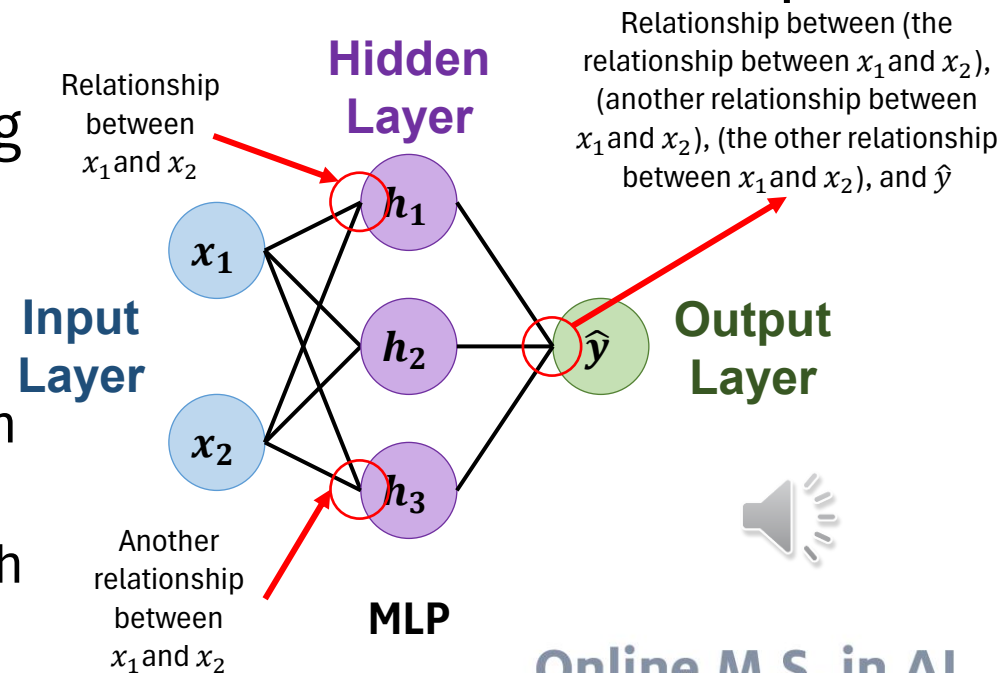
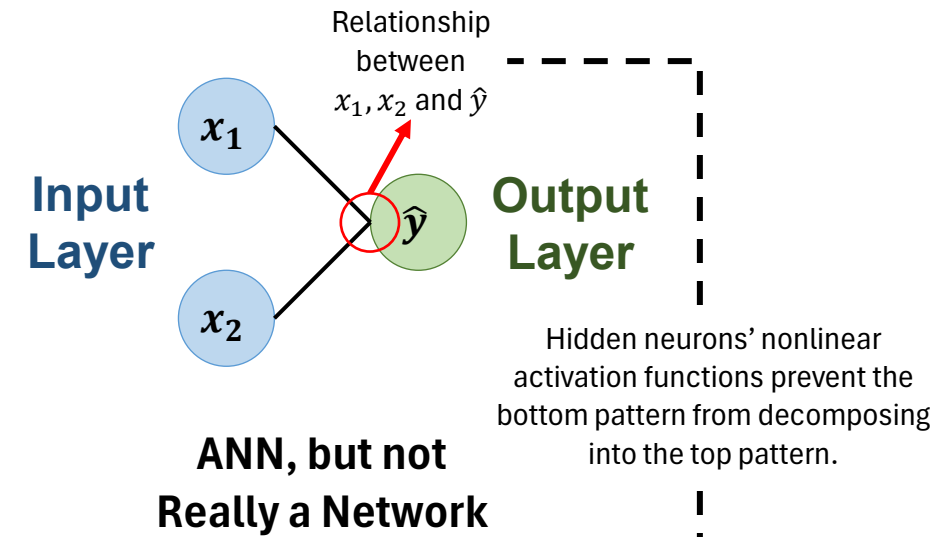
Artificial Neural Networks

- ▶ Weighted sum alone means learning only linear patterns.
- ▶ But not all real-world problems are linear.
 - ▶ **Recall from Week 2:** SVMs learning non-linear decision boundaries with non-linear kernel functions!
- ▶ Activation functions add the “warping” that allows the network to learn complex, non-linear shapes.



Artificial Neural Networks

- ▶ How can an artificial neuron learn complex nonlinear patterns with just an activation function?
 - ▶ Is the human brain made of a single neuron?
- ▶ The complexity of ANNs comes from connecting multiple neurons together in layers to form a **Multilayer Perceptron (MLP)**.
 - ▶ **Feedforward algorithm:** Output of one layer's neuron is used as the input for connected neurons in the next layer.
 - ▶ Information flows “forward” from input layer, through fully-connected layers, and out the output layer.



Artificial Neural Networks (ANNs)

- ▶ Different activation functions are adopted for neurons in different layers (all nonlinear):
 - ▶ **Hidden layer** to help the model learn complex, non-linear patterns.
 - ▶ **Output layer** to match the format of the answer we need

Suitable Layer	Function	Formula	Output Value Range	Characteristics	Common Tasks
Hidden	ReLU	$\max(0, x)$	$[0, \infty)$	Extremely fast to compute and helps deep networks learn efficiently	General Use
	Tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$(-1, 1)$	Zero-centered; usually performs better than Sigmoid in hidden layers.	General Use
Output	Linear	x	$(-\infty, \infty)$	Passes the input directly to the output without changes.	Regression & Time Series
	Sigmoid	$\frac{1}{1 + e^{-x}}$	$(0, 1)$	Perfect for predicting a single probability	Binary Classification
	Softmax	$\frac{e^{x_i}}{\sum_j e^{x_j}}$	$(0, 1)$	Ensures all output probabilities sum up to exactly 1.0.	Multi-Label Classification



ANNs (*Overview*)

- ▶ Taken together, ANNs can be described as such:
 - ▶ **Input:** Datapoints with a label and v features, with each feature having its own input neuron (input layer of size v).
 - ▶ **Parameters:** All the weights inside the network
 - ▶ **Initialization:** Weights are randomized.
 - ▶ **Output (Feedforward Algorithm):** Final layer (the output layer) has one neuron for each possible label category. Each output neuron's value is the calculated probability of its corresponding label.
 - ▶ Predicted label is the label with the highest corresponding output neuron value.



ANNs (Overview)

- Taken together, ANNs can be described as such:

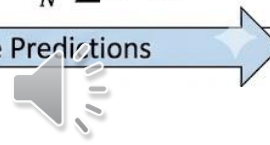
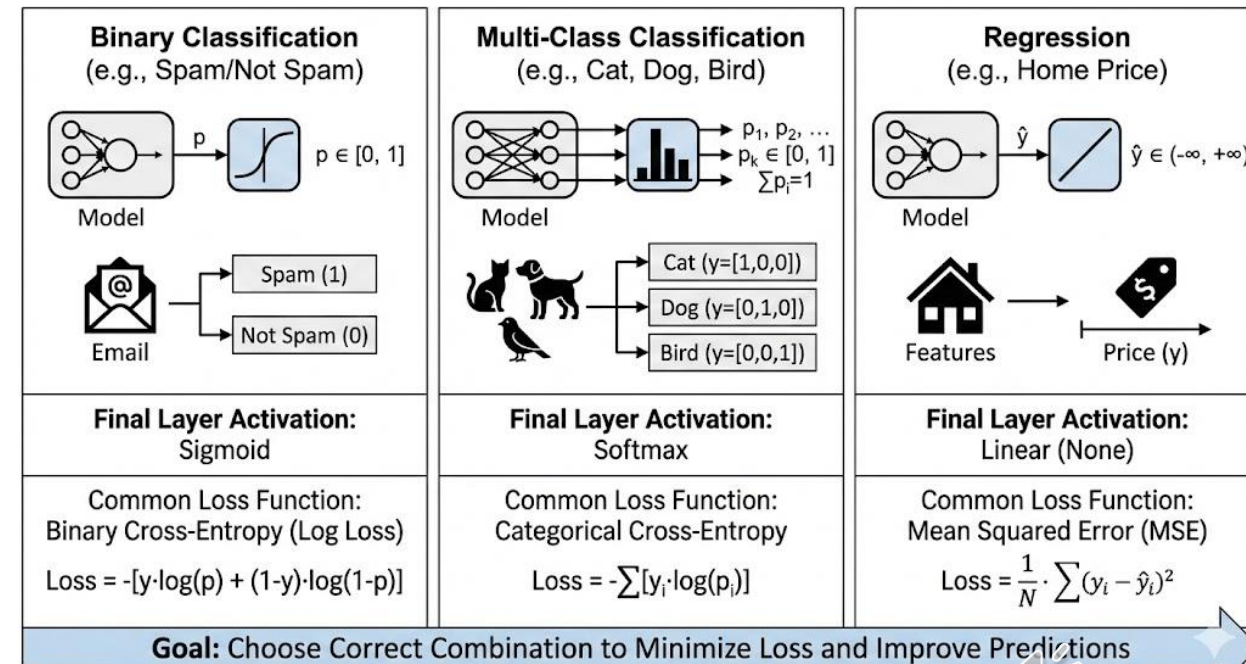
- Loss Function:** Measures how different the predicted label ($\hat{y}$) is from the real label (y).

- Exact function varies on the task at hand!

- Optimization (Backpropagation Algorithm):** Adjust each weight according how much the weight “contributed” to the loss.

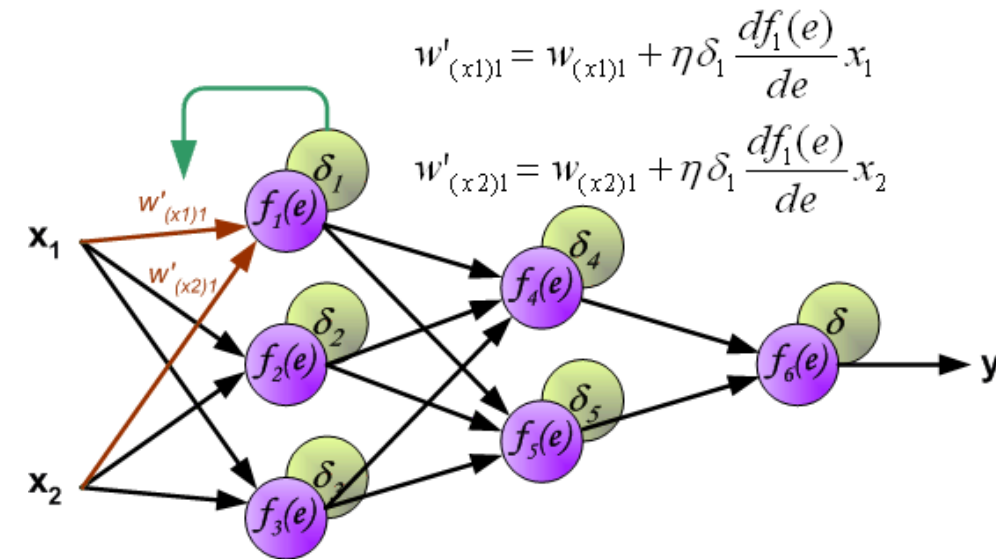
- Converge:** Loss no longer decreases with additional training steps.

- Core Intuition:** Imitate how the brain (one of nature’s best learning machines) learns with its neurons.



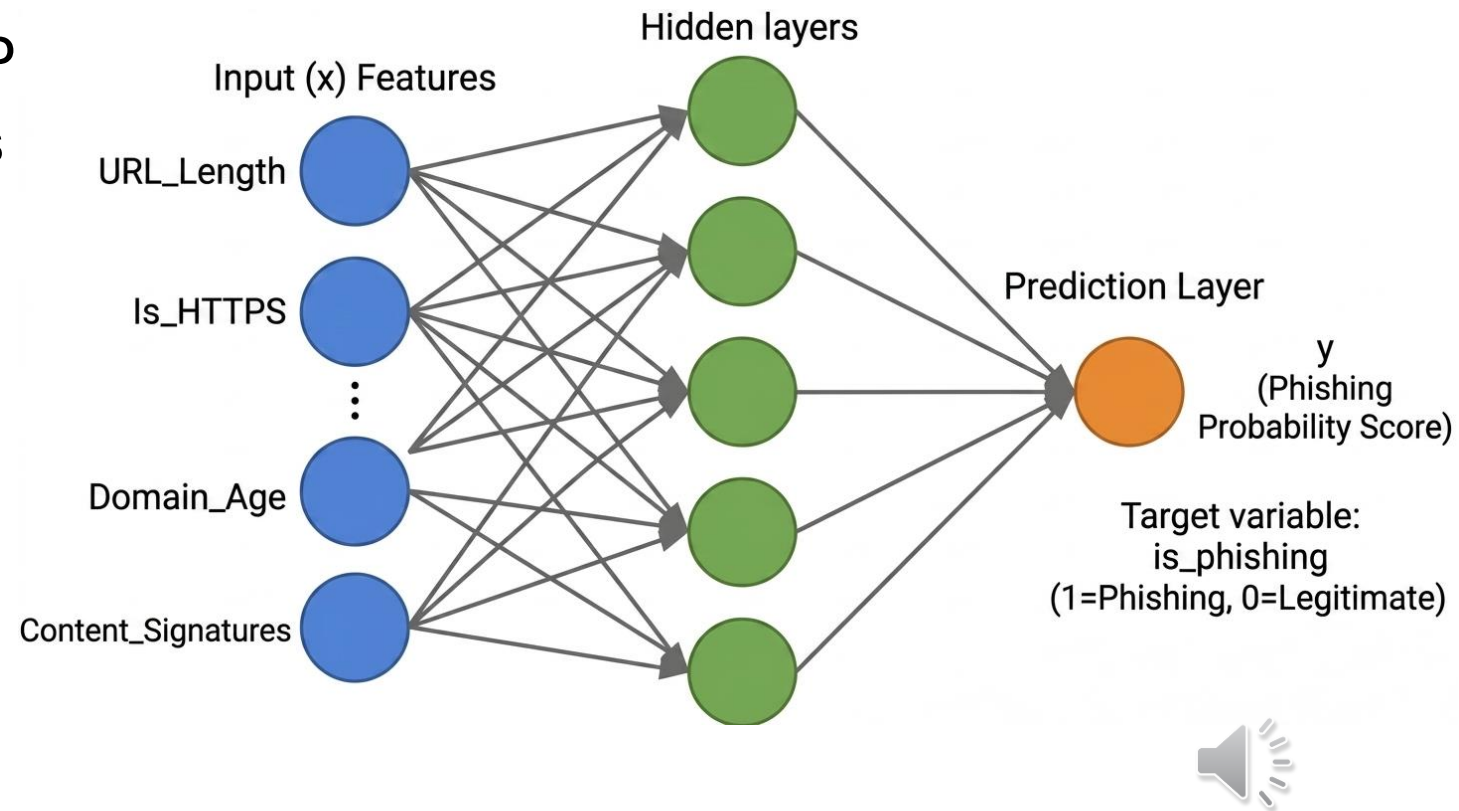
ANNs

- ▶ **Backpropagation Algorithm** mathematical describes how much each weight should be adjusted (Rumelhart et al. 1986):
 - ▶ Calculate how much each weight contributed to the error at each step (δ_n).
 - ▶ Adjust the weights gradually by going backwards with **gradient descent**.
- ▶ Feedforward and Backpropagation are the two fundamental mathematical processes that describe how ANNs operate and learn.



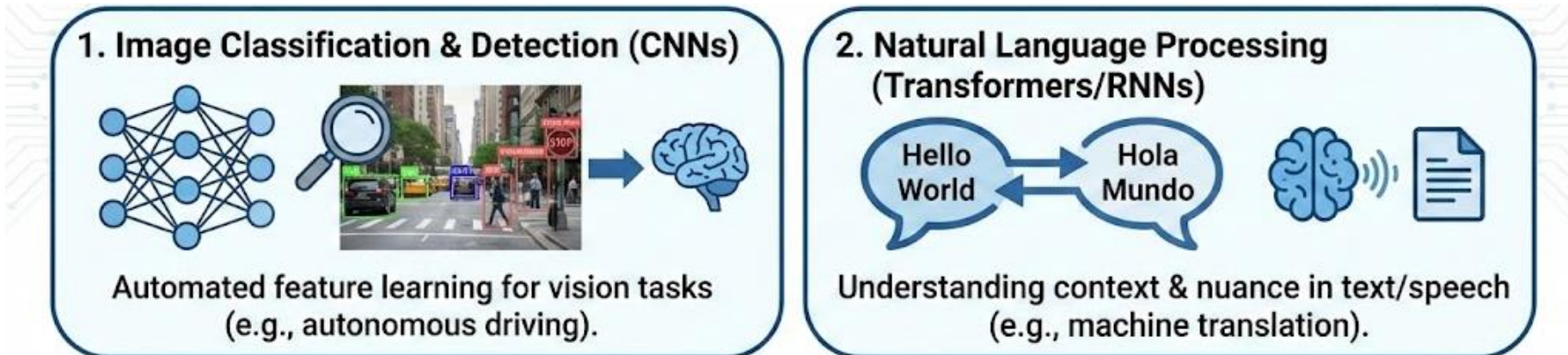
ANNs – Python Example

- ▶ **Python example:** Phishing Detection (revisited) using MLP
 - ▶ Refer to supplemental materials
- ▶ **Data:** Prior phishing websites with 50+ different features per website
- ▶ **Key Packages:** PyTorch



ANNs

- ▶ The ANN serves as the foundation for the Deep Learning (DL) renaissance we find ourselves in today.
 - ▶ The deep in DL refers to multiple hidden layers learning increasingly specific sub-patterns in the general pattern that maps the input to the output.
 - ▶ MLPs are the simplest DL model as they have one hidden layer.
- ▶ Most modern AI models are improvements upon MLPs, adjusting how neurons operate and how layers are connected to each other to better learn more complicated patterns.

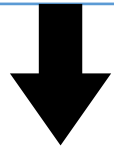


Convolutional Neural Network (CNN)

- ▶ An example of more complicated patterns can be found in images.

- ▶ Recall from Week 1: Individual pixels are tokenized into lists of numbers.

Data Type	Tokenization Procedure	Example Untokenized Datapoint	Example Tokenized Datapoint
Images	Leverage each pixel's red, green, and blue values		[112, 173, 71], [237, 125, 49], [112, 48, 160]



- ▶ Standard MLPs often treat images as a long, flattened 1-D input.
 - ▶ Struggle to learn positional patterns, which are important for images.

[112, 173, 71, 237,
125, 49, 112, 48, 160]

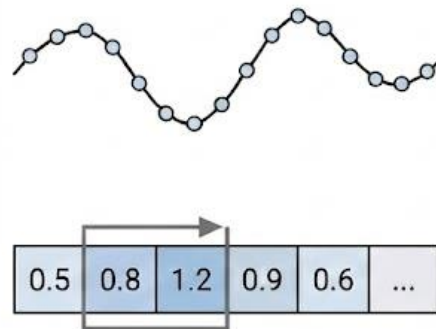
- ▶ Is there a way to edit how neurons are connected/calculated to reflect the spatial topology of image data?



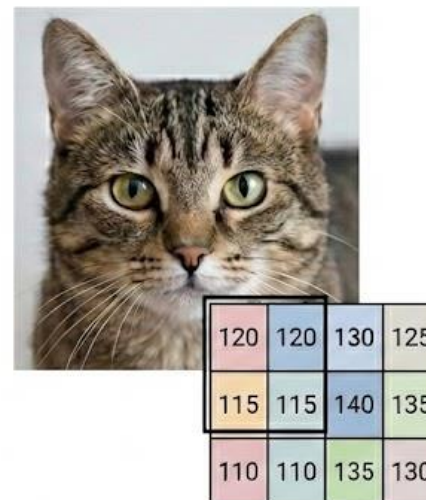
CNN

- ▶ **Convolutional Neural Networks** (or Convolutional Networks, or CNNs, or ConvNets) can process data with a **grid-like or array** topology
 - ▶ 1-D grid: time-series data, sensor signal data
 - ▶ 2-D grid: image data
 - ▶ 3-D grid: video data

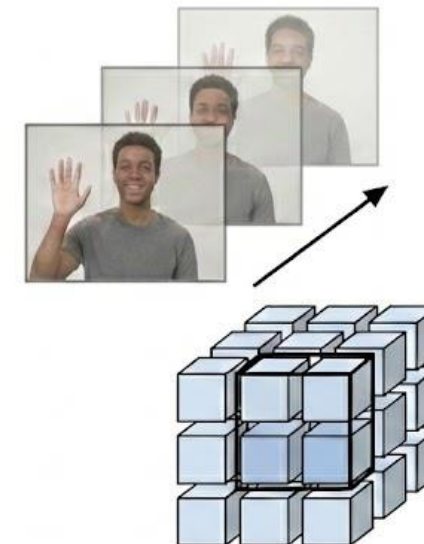
1-D GRID: Time-Series / Sensor Data



2-D GRID: Image Data

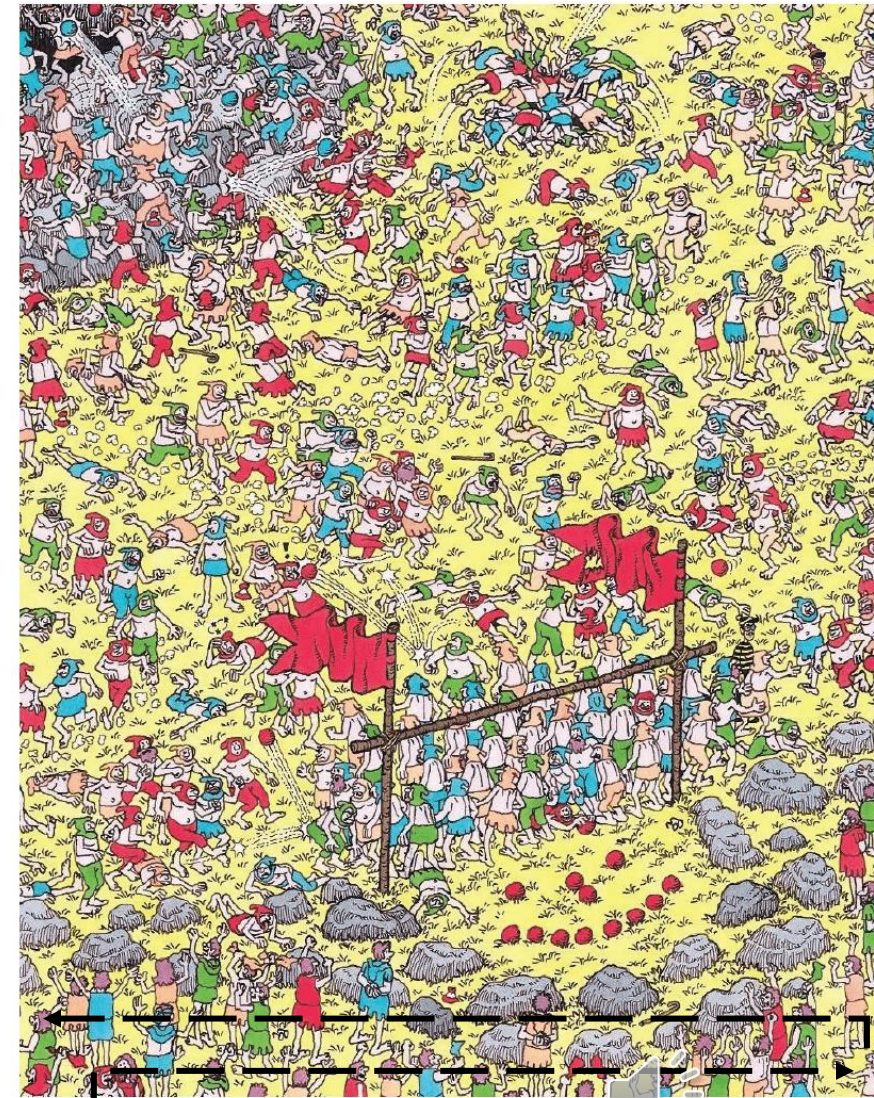


3-D GRID: Video Data



CNN (Overview)

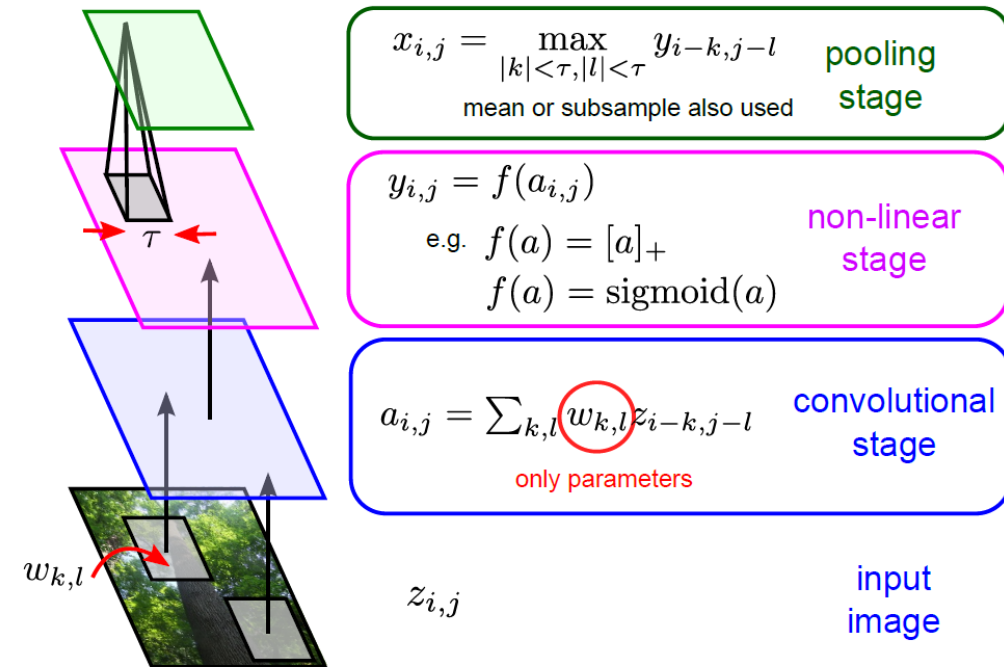
- ▶ CNNs are ANNs improved to handle array topology:
 - ▶ **Input:** Same as ANNs, but **more applicable to images and spatial data**.
 - ▶ **Output and Parameters :** Same as ANNs, **but now with convolution layers!**
 - ▶ **Loss Function, and Optimization:** Same as ANNs
- ▶ **Core Intuition:** When you are looking for Waldo, do you scan each pixel to see if they form a Waldo?
 - ▶ Instead, you most likely slide a “Waldo pattern” across the image to see if there are any matches.



This is your “filter” as you convolute the image

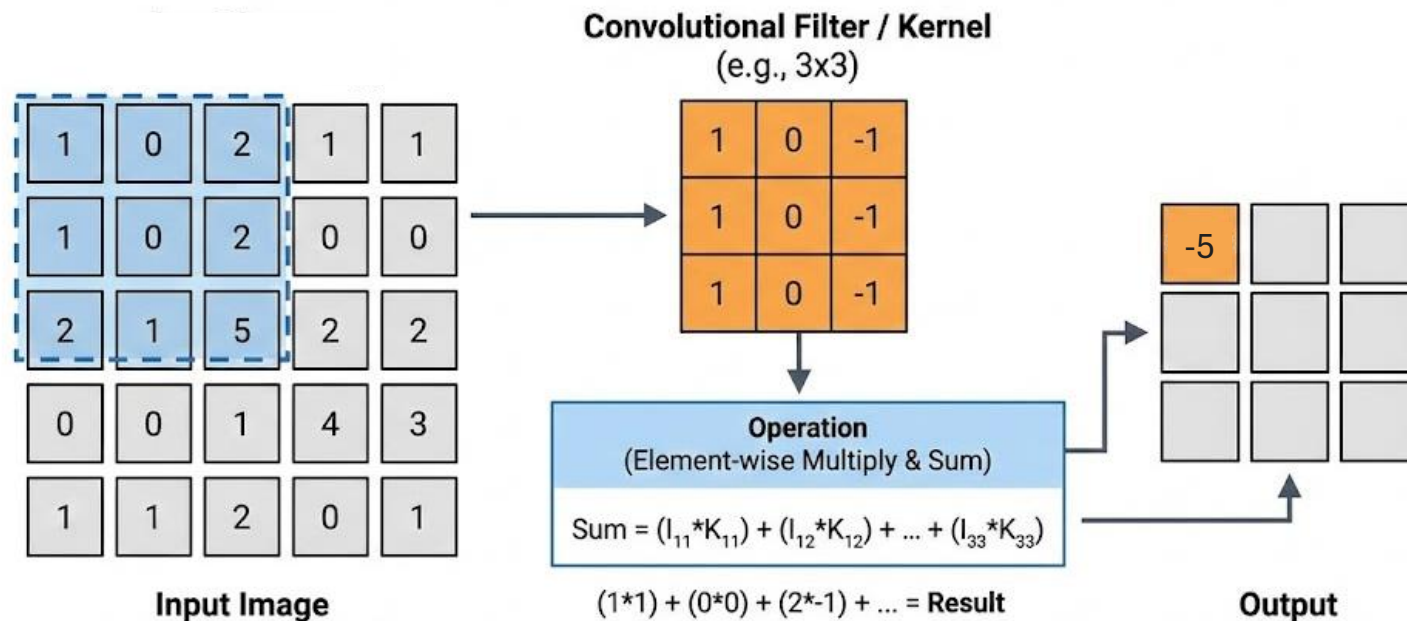
CNN – Convolution Layers

- ▶ The core improvement CNNs offer is the convolution layer, consisting of 3 major stages:
 - ▶ **Convolutional**: Detecting local features through filters.
 - ▶ **Non-linearity**: Normalization via Rectified Linear Unit (ReLU).
 - ▶ **Pooling**: Merging similar features.
- ▶ Compared to ANNs' fully-connected layers, convolution layers learn topological patterns with less parameters → efficient learning



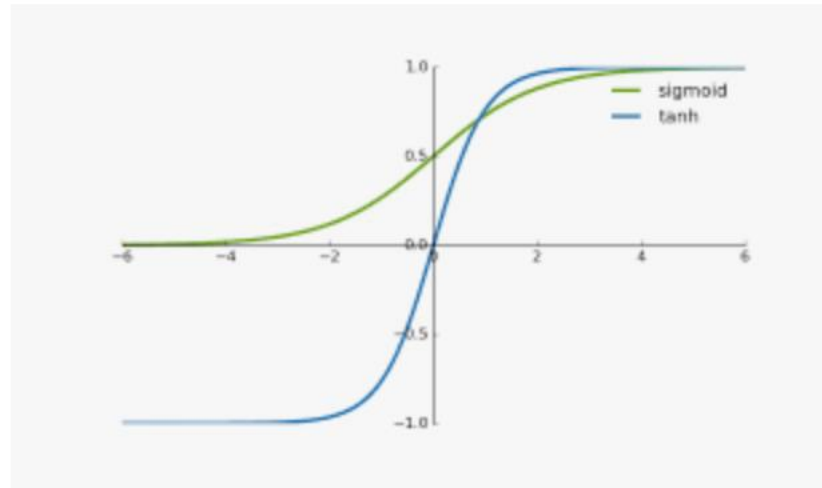
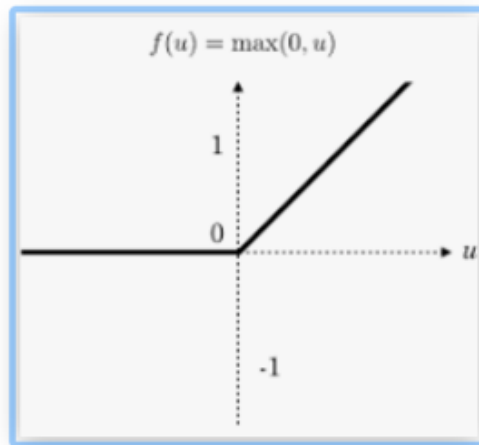
■ CNN – Convolution Layers (Convolutional Stage)

- ▶ **Goal:** Extract topological patterns from the input arrays (e.g., images).
- ▶ **Mechanism:** Slide a **filter** across the input data to perform element-wise multiplication and summation.



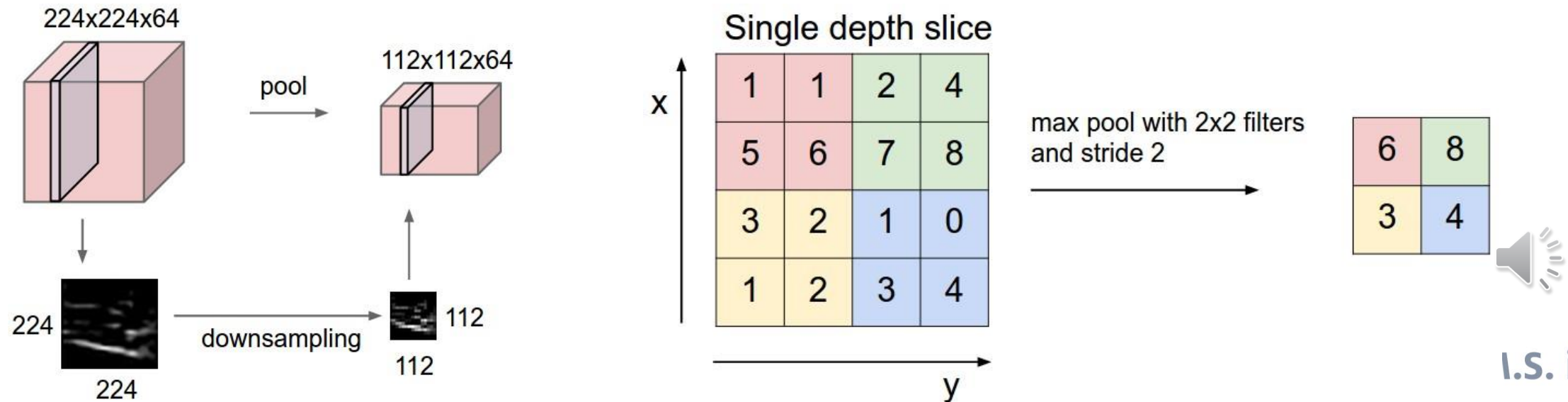
■ CNN – Convolution Layers (Non-Linear Stage)

- ▶ **Goal:** Increase the nonlinearity of the entire architecture
- ▶ **Mechanism:** Applies non-linear activation functions to the convolution outputs



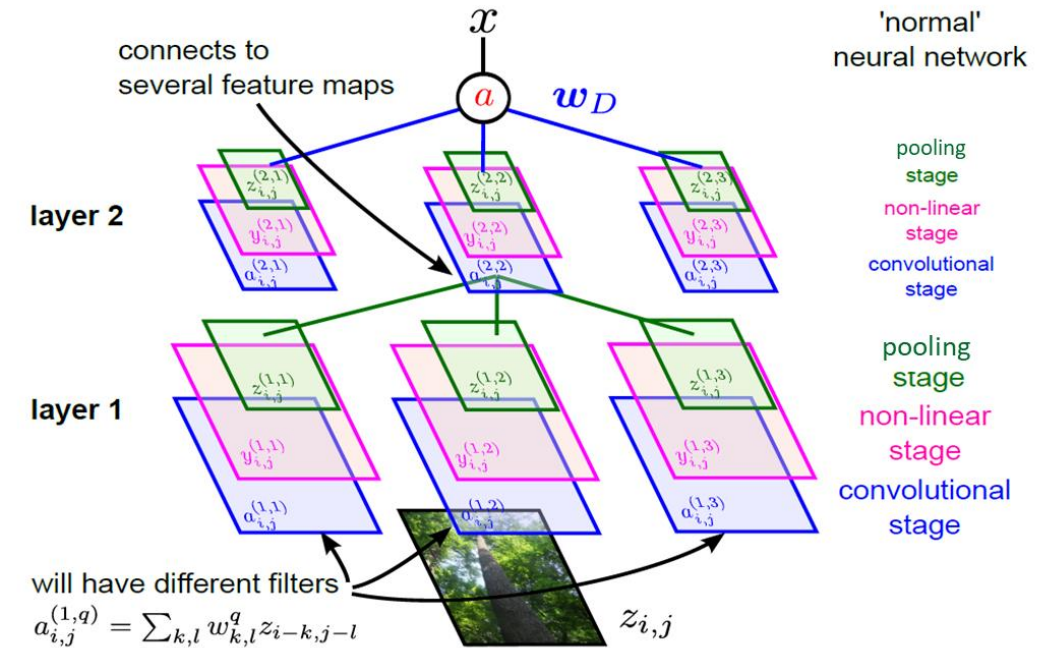
CNN – Convolution Layers (Pooling Stage)

- ▶ **Goal:** Reduce the size of the representation
 - ▶ Saves on required computation to train.
- ▶ **Mechanism:**
 - ▶ Partition input into non-overlapping sub-regions and aggregates each region's value.



CNN

- ▶ CNNs can have multiple convolution layers to learn different levels of the spatial pattern.
 - ▶ Each layer can have multiple filters that learn different spatial patterns at the same level.
- ▶ Recent CNN architectures have 10-20 convolution layers.
- ▶ After a few sets, the information is typically sent to one or two **fully connected** layers before outputting.
 - ▶ The typical activation function is the **sigmoid** function.
 - ▶ Output is typically class (classification) or real number (regression).



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

CNN – Cybersecurity Applications



1. OBJECT DETECTION (Locate & Classify)

THREAT ACTOR PROFILING:
Identify Specific Attackers &
Classify Groups



2. IMAGE SEGMENTATION (Pixel-level Classification)

NETWORK TRAFFIC SEGMENTATION:
Isolate Anomalous Packets &
Identify Malware Headers



3. FACIAL RECOGNITION (Identify & Verify)

IDENTITY & ACCESS MANAGEMENT (IAM):
Secure Multi-Factor Authentication
& Biometric Verification



4. ANOMALY DETECTION (Identify Deviations)

SECURITY EVENT & LOG ANALYSIS (SIEM):
Detect Deviant System Behaviors
& Potential Breaches



Recurrent Neural Networks (RNN)

- ▶ Another datatype that can hide more complicated patterns is language.
 - ▶ The meaning of a feature in textual data (e.g., an individual word) depends on the features that appear before it (sequential dependencies).
- ▶ Standard MLPs often cannot represent the sequential nature of textual data.
- ▶ Is there a way to also input a representation of prior features?

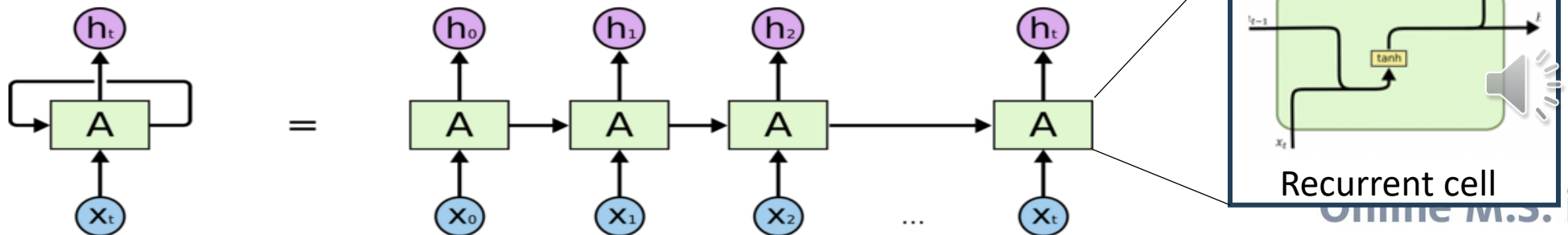
Same feature,
but different
meanings due
to sequential
dependencies

On my walk next to the **river**
bank, I remembered that I need
to **cash** a check. Thus, I ended
by **river**-side walk to head for a
bank.



RNN (Overview)

- ▶ **Recurrent Neural Networks (RNN)** are networks with loops, allowing information to persist:
 - ▶ **Input:** Same as ANNs, but **also the hidden layers of the previous training step.**
 - ▶ **Parameters, Output, Loss Function, and Optimization :** Same as ANNs.
- ▶ **Core Intuition:** You interpret a word based on an understanding of the text that was built by several “past yous” reading previous words.
 - ▶ RNN can be thought of as multiple copies of the same network, each passing their knowledge to a successor.





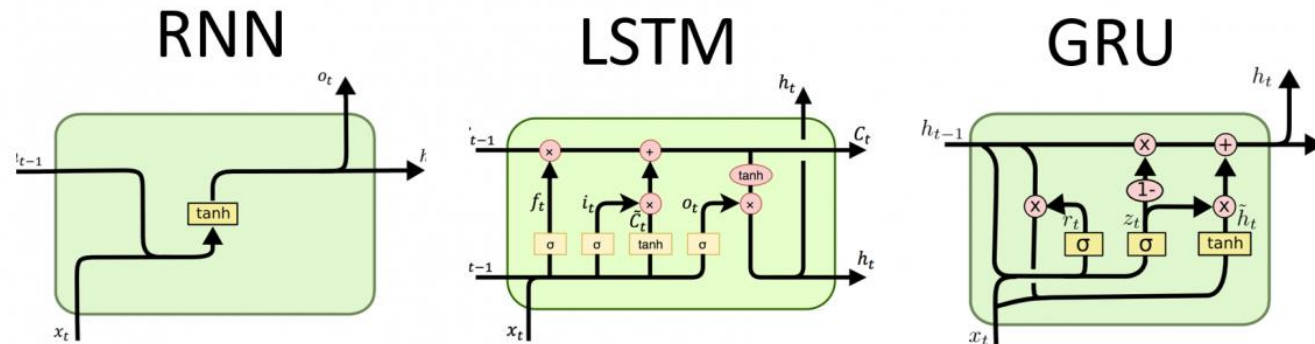
RNN

- ▶ The recurrent structure of RNNs exhibits the following characteristics:
 - ▶ Specialized for processing a sequence of features $x_1, \dots, x_\tau$
 - ▶ Each feature x_i is processed with the **same network A that preserves past information**
 - ▶ Can scale to much **longer sequences**.
 - ▶ Reusing network **A** reduces the required amount of parameters in the network
 - ▶ Can process **variable-length sequences**.
 - ▶ The network complexity does not vary when the input length change.



RNN – Advanced Variants

- ▶ RNN passes information in a “multiplicative” way. Can often result in:
 - ▶ **Vanishing gradient:** Past information fades away, preventing early layers from learning.
 - ▶ **Exploding gradient:** Past information grows chaotic, causing the model to crash.
 - ▶ Too unstable to learn long-term dependencies
- ▶ Two “additive” recurrent cell designs were proposed and widely adopted:
 - ▶ **Long Short-Term Memory (LSTM)** (Hochreiter and Schmidhuber, 1997)
 - ▶ **Gated Recurrent Unit (GRU)** (Cho et al. 2014)



References:

- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724-1734). Doha, Qatar: Association for Computational Linguistics..
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.



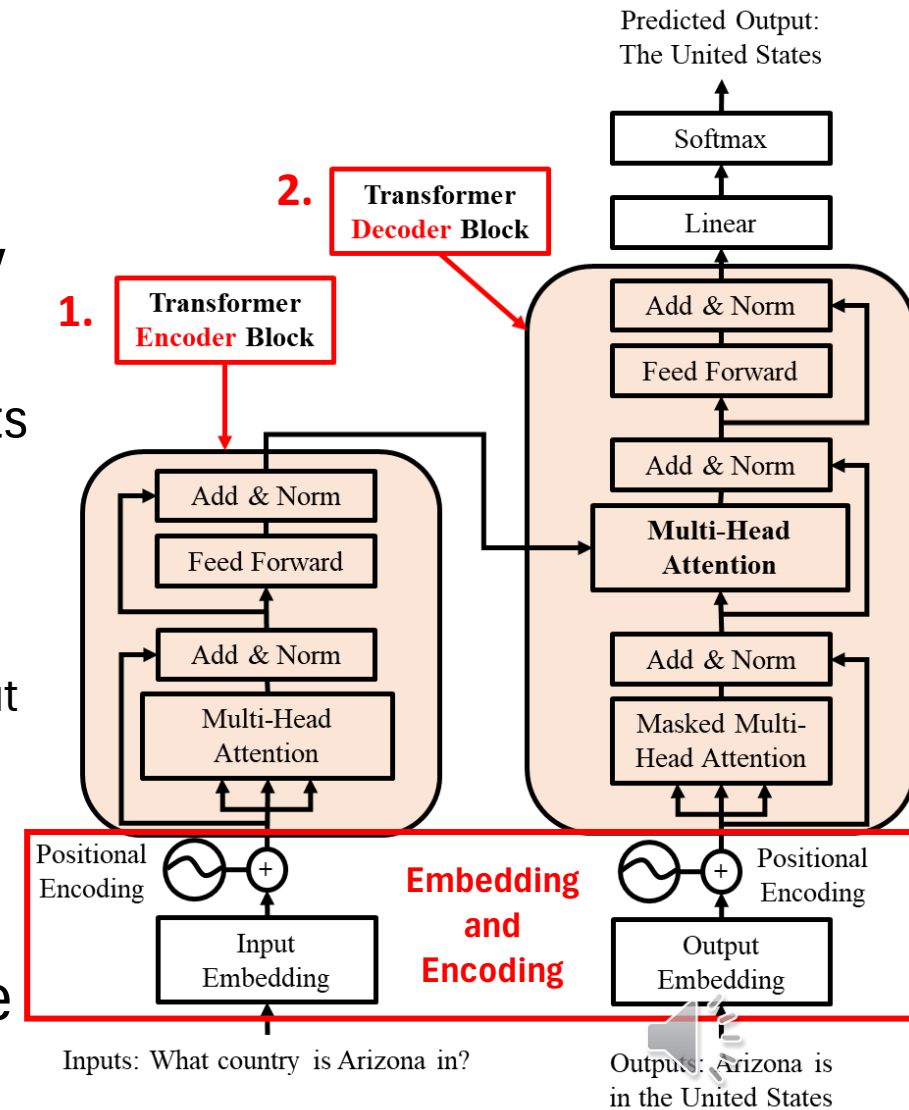
Transformer

- ▶ Prior to 2017, **RNN** architectures (LSTM, GRU) had seen **state-of-the-art** performance for various **natural language processing** (NLP) tasks.
- ▶ However, **sequential computation** was still **infeasible** for long sequences.
 - ▶ Long recurrence tend to **forget** information from the beginning of the sequence.
- ▶ Instead of sequentially reading input, can we read the input all at once?
- ▶ In 2017, the Google Brain research team noted the success of **attention mechanisms** and built a new model known as the **Transformer** (Vaswani et al. 2017).
 - ▶ This model has no convolutions or recurrence.
 - ▶ The Transformer implements **attention mechanisms** to achieve state-of-the-art NLP performance.



Transformer (Overview)

- ▶ **Transformer** models are designed to process every part of input sequence simultaneously.
 - ▶ **Input (Embedding and Encoding)**: Sequential datapoints tokenized with positional encodings
 - ▶ **Output**: Two major components:
 1. **Encoder Block**: Create latent representation of text
 2. **Decoder Block**: Decode latent representation for task output
 - ▶ **Loss Function**: Typically, the cross-entropy loss
 - ▶ **Parameters, and Optimization** : Same as ANNs.
- ▶ **Core Intuition**: Interpreting a word does not require re-reading the whole text, just paying attention to the relevant prior words.



Transformer – Embedding and Encoding

1. Tokenization

Write a story.

Write A story .

Tokenization: Turning words into tokens

1. Textual inputs are first **tokenized**, transforming sentences into individual tokens.

2. Embedding

Write	2.13	-0.42	...	-1.03
A	0.91	0.56	...	0.23
story	-1.56	1.34	...	0.14
.	1.42	-1.32	...	-2.41

2. Tokens are **transformed** into **vectors** based on co-occurrence statistics (e.g., Word2Vec).

3. Positional encoding

Write	2.13	-0.42	...	-1.03	+		Write (1)	
a	0.91	0.56	...	0.23	+		a (2)	
story	-1.56	1.34	...	0.14	+		story (3)	
.	1.42	-1.32	...	-2.41	+		.(4)	

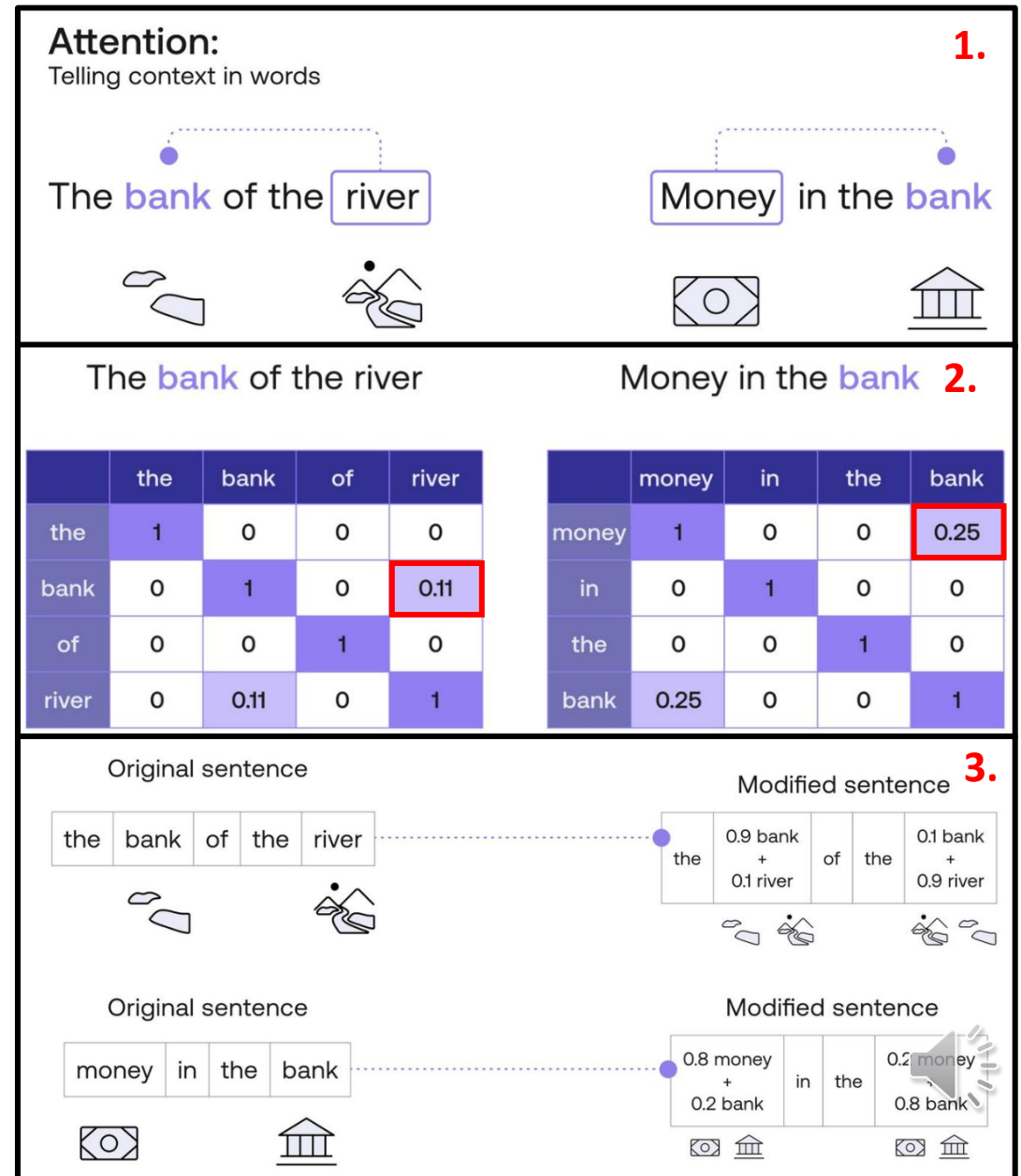
3. The **position** of each **token** in the sentence is added to the vector, such that sentences with the same words in a different order will have different vectors.

4. The vectors with added positional encoding are the high-dimensional **word embedding** that is inputted into the Transformer model.

- Recall from SVM's Kernels: Higher dimensionality can represent more complex patterns (such as sequential nature of text).

Transformer – Encoder and Decoder Block

- ▶ A key component in the encoder and decoder is the **attention mechanism**.
- ▶ The word “bank” has numerous meanings depending on the context it is used.
 - ▶ We need context for the task.
- ▶ Attention helps identify context by determining the importance of nearby words.

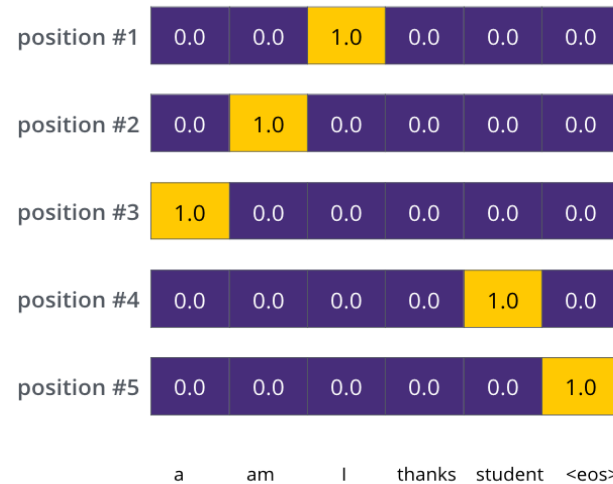


Transformer – Loss Function

- ▶ The transformer uses a simple **cross-entropy loss** to compare the target outputs and the predicted outputs.

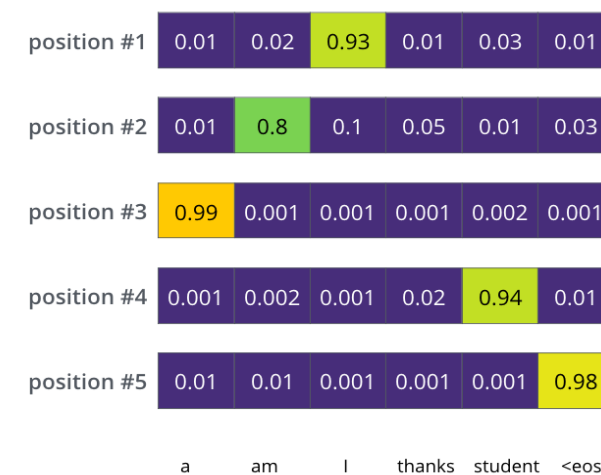
Target Model Outputs

Output Vocabulary: a am I thanks student <eos>



Trained Model Outputs

Output Vocabulary: a am I thanks student <eos>



(from <https://jalammar.github.io/illustrated-transformer/>)





Transformer

- ▶ Transformers are originally designed for sequence-to-sequence tasks (e.g., machine translation, summarization).
- ▶ However, real-world text is massive, dense, and uses highly specialized terminology.
 - ▶ E.g., Dark-web markets could have millions of posts with specialized terminologies for illicit goods/services and unique hacker lingo.
- ▶ We need trillions of words in the training data and millions of dollars in computational resources just to teach the model basic grammar.
 - ▶ Unfortunately, not all of us can be “tony-stark-rich.”



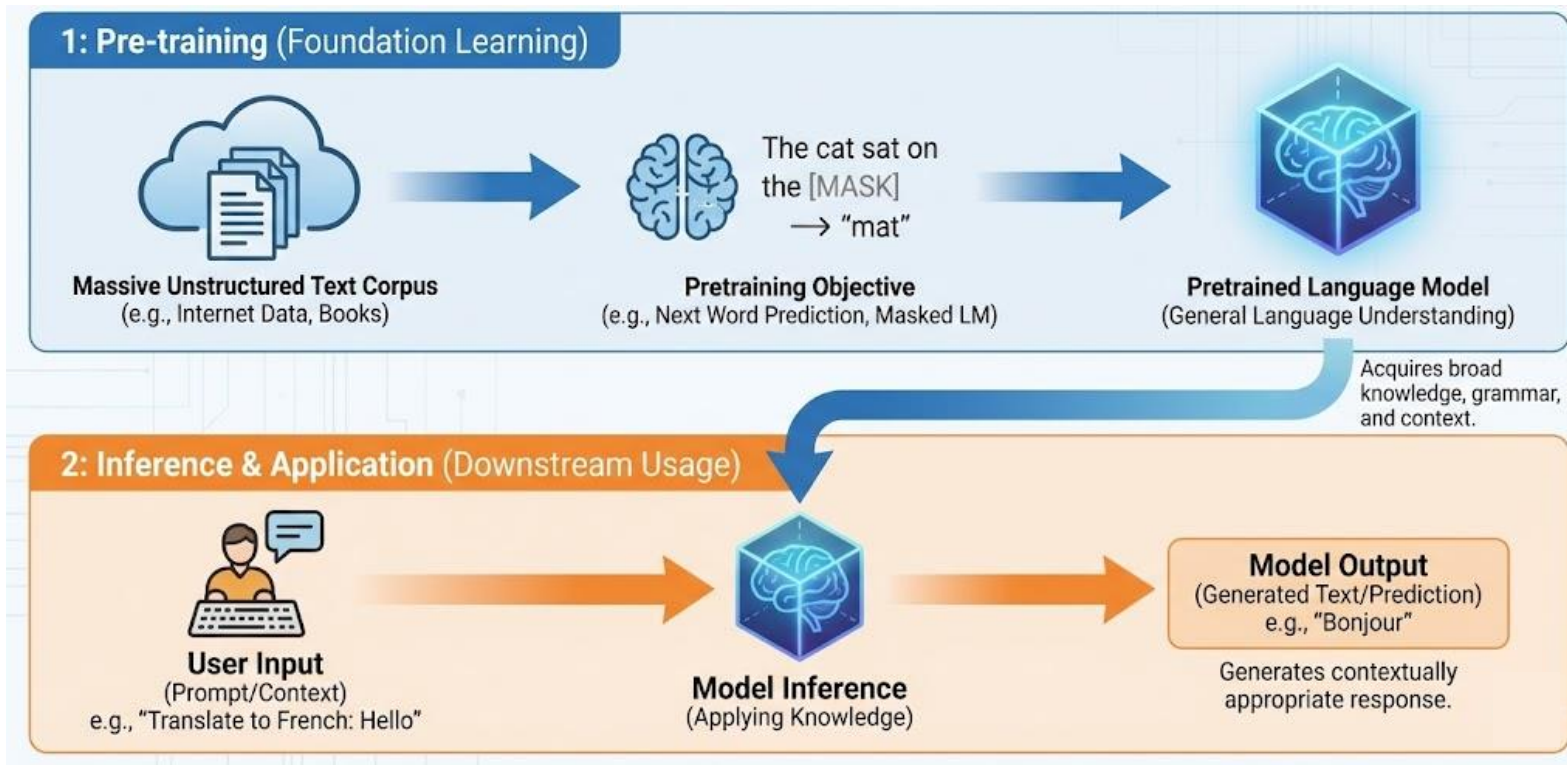
Pretrained Language Models

- ▶ Starting in 2018, the field evolved from training task-specific models toward *pre-training* foundational architectures.
 - ▶ These models are trained on massive, unlabeled datasets to capture general linguistic patterns.
 - ▶ Efforts have been made in specializing Transformer's architecture (e.g., **Encoder** for deep bidirectional understanding, **Decoder** for generative scaling).
- ▶ In **2022**, the integration of human feedback unlock the **ChatGPT era**, producing models capable of following complex instructions and conversational reasoning.



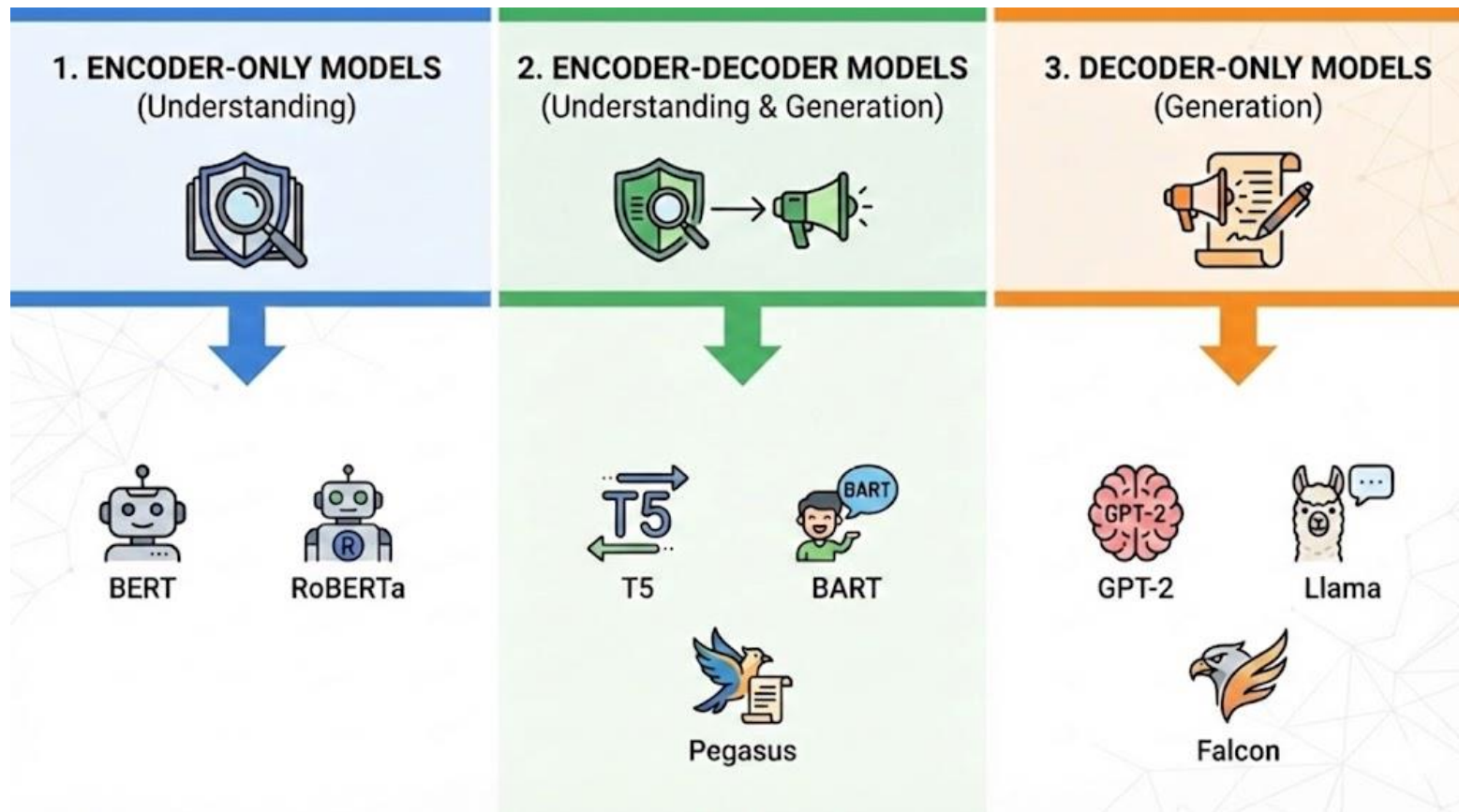
Pretrained Language Models

- Instead of starting from zero, models are **pretrained** with natural languages (e.g., news, Wikipedia) to understand how human languages work.
 - E.g., when the model sees term “social”, the next word is likely to be “media.”



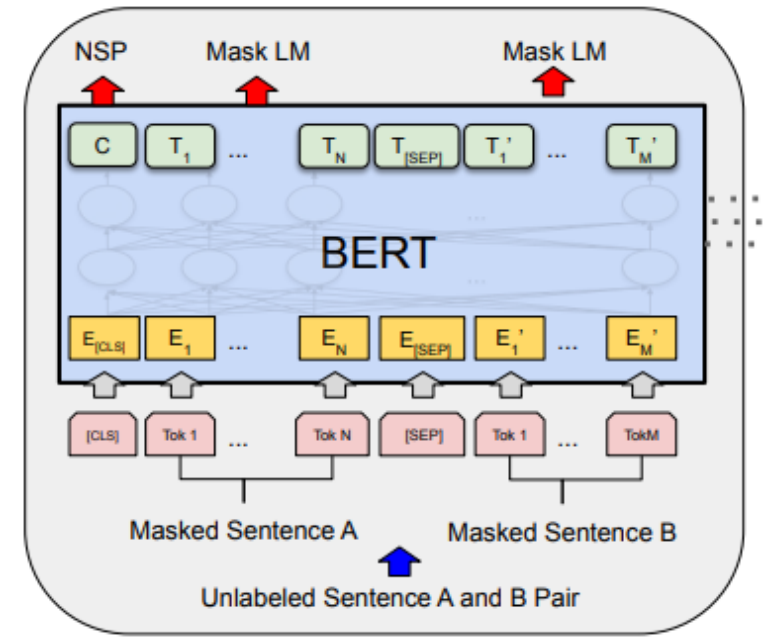
Pretrained Language Models – Categories

- ▶ **Pretrained language models (PLMs)** can be divided into three categories based on the components adopted from the transformer.



Pre-Trained Language Models – BERT

- ▶ BERT is a fundamental encoder-only model released in 2018 by **Google's AI Research team**.
- ▶ BERT is pre-trained on the **BooksCorpus** (74 million sentences from 11,038 books) and **English Wikipedia** (2.5 billion words).
- ▶ Two-pretraining objectives:
 - 1. Masked Language Modeling (Mask LM)**
 - ▶ A random word in a sentence is masked (i.e., hidden) from the model.
 - ▶ The model is trained to predict the masked word in the sequence.
 - 2. Next Sentence Prediction (NSP)**
 - ▶ The training dataset is split into two-sentence pairs
 - ▶ The model is trained to predict if the second sentence follows the first.



BERT Pre-Training
(from Devlin et al., 2018)

Mask token: [MASK]

I am a [MASK] student.

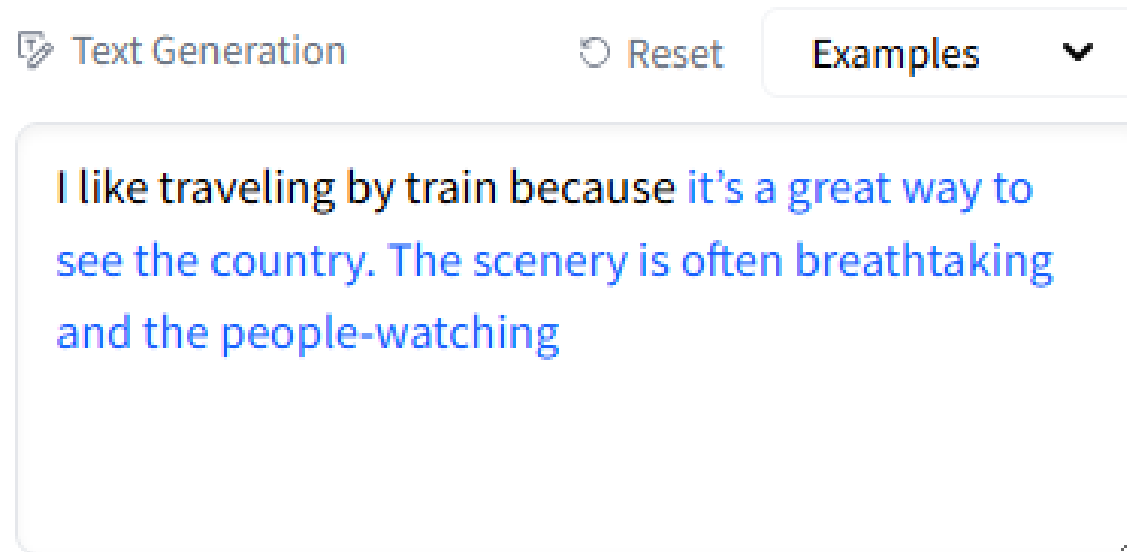
Generate

college	0.147
good	0.086
law	0.084
graduate	0.078

BERT Inference (from huggingface)

Pre-Trained Language Models – GPT

- ▶ **GPT** was introduced in **2018** by **OpenAI** as a paradigm-shifting architecture that uses an **autoregressive** training strategy (Radford et al., 2018).
 - ▶ GPT is pre-trained on the **BooksCorpus** (74 million sentences from 11,038 books).



Text Generation Inference (from huggingface)

Pre-Trained Language Models – GPT

- ▶ **Autoregressive Training:** Generate future values based on past values.
 - ▶ The goal of GPT is to **maximize** the **conditional probabilities** of each predicted token given prior tokens as context.
 - ▶ Text “Generation” is just iterative next token classification.
 - ▶ The predicted token is appended to the text and used as the next step’s input.

Text: Second Law of Robotics: A robot must obey the orders given it by human beings

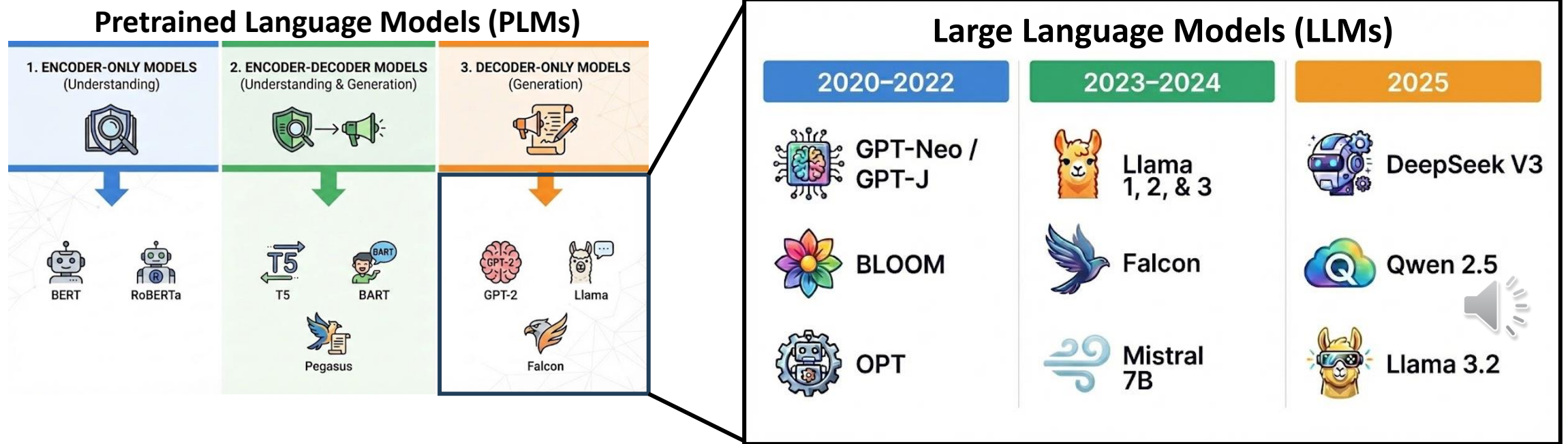
Generated training examples

Example #	Input (features)	Correct output (labels)
1	Second law of robotics :	a
2	Second law of robotics : a	robot
3	Second law of robotics : a robot	must
...		



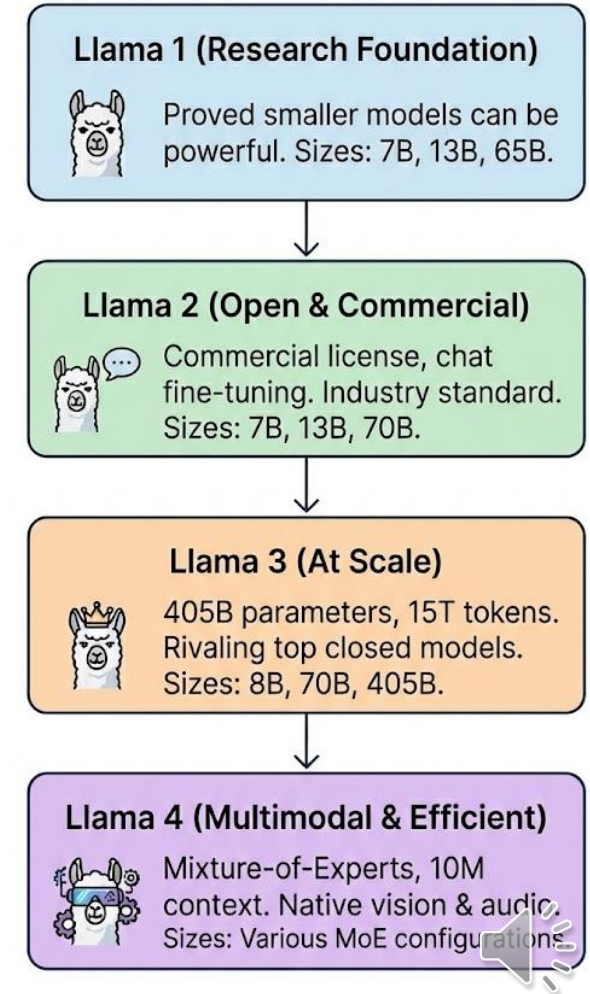
Large Language Models

- ▶ Recent PLM development have been largely focused on **decoder-only** Generative AI (GenAI) models.
- ▶ As these models scale to billions of parameters, they exhibit emergent reasoning capabilities and are known as **Large Language Models (LLMs)**.



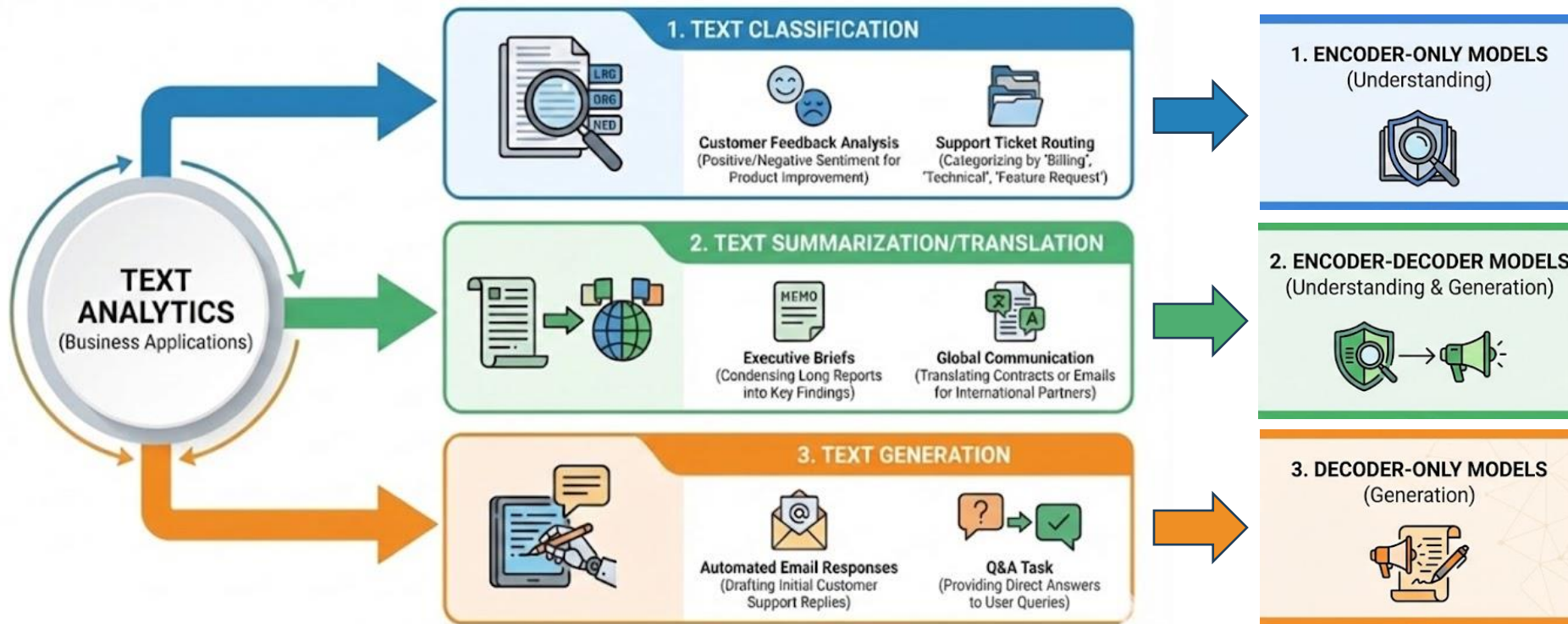
Large Language Models – Llama

- ▶ **Llama** is developed by Meta and is one of the most popular open-source LLM.
 - ▶ The model parameters are available on HuggingFace.
 - ▶ Developer can adapt it for specific tasks (e.g., advanced coding, financial statement analysis).
- ▶ It matches the intelligence of top-tier closed-source models (like GPT-4) while being significantly more efficient and cheaper to operate.
- ▶ Starting from Llama 3, it can now process images, understand charts, and handle massive amounts of data (like entire books).



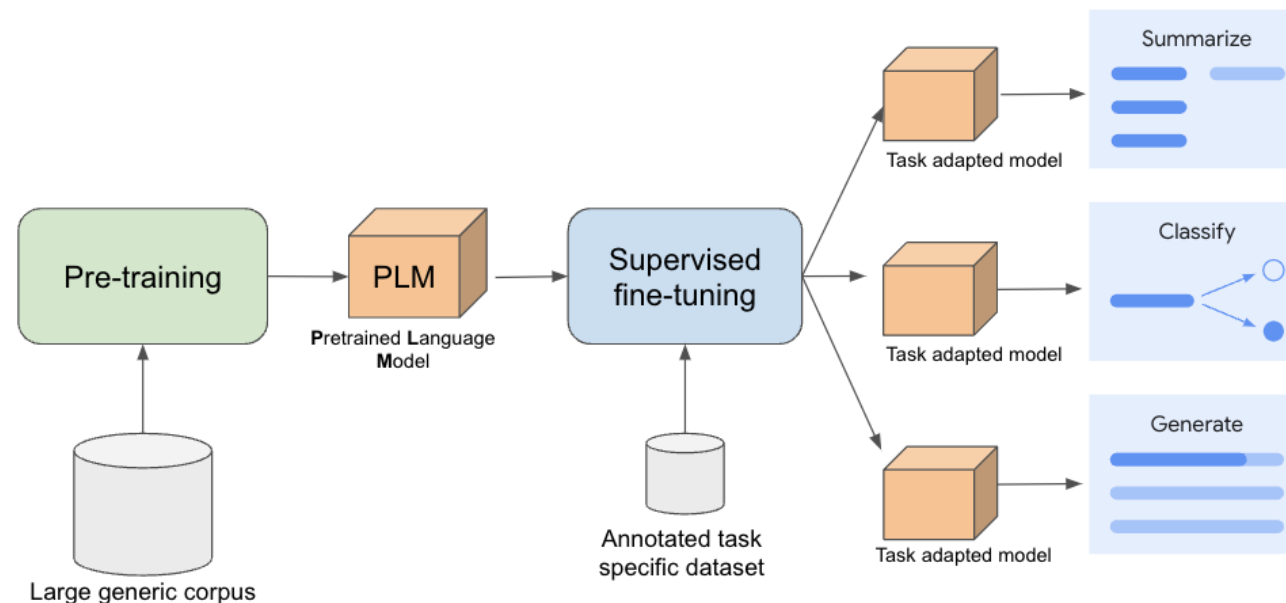
Task Adaption

- ▶ Pretraining alone is usually not sufficient for all kind of text analytic tasks.
 - ▶ Pretrained models also don't understand domain-specific terminologies.
- ▶ **Task adaption** aim to use additional training data to allow PLMs to perform effectively on specific domains and tasks.



Task Adaption – Supervised Fine-Tuning

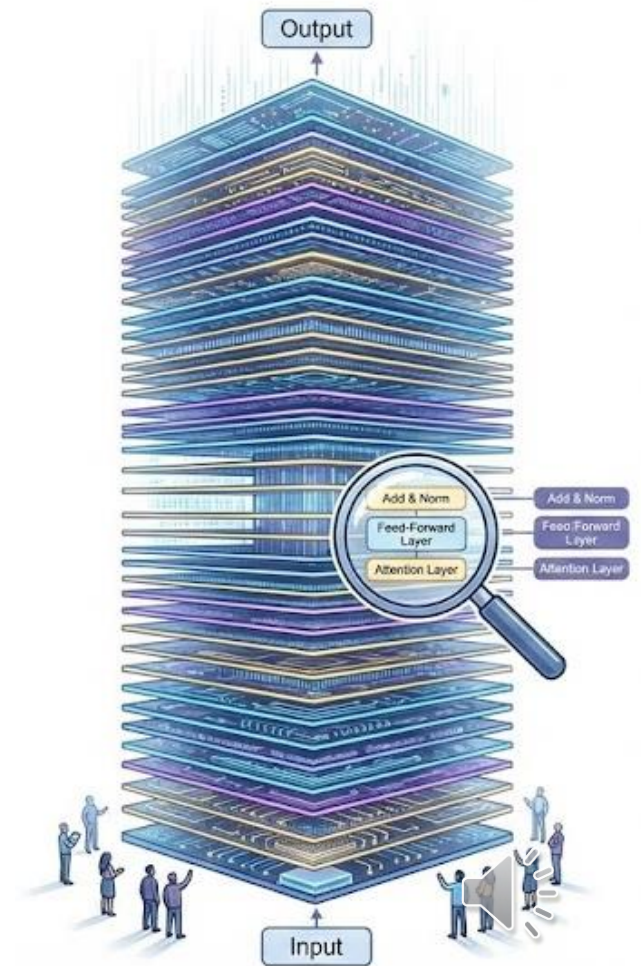
- Supervised fine-Tuning (SFT) is the most straightforward method for task adaptation.
- **Core Intuition:** Use additional training data to slightly update the parameters of pre-trained model (W) to make it task specific (W').



(From <https://cloud.google.com/blog/products/ai-machine-learning/supervised-fine-tuning-for-gemini-llm>)

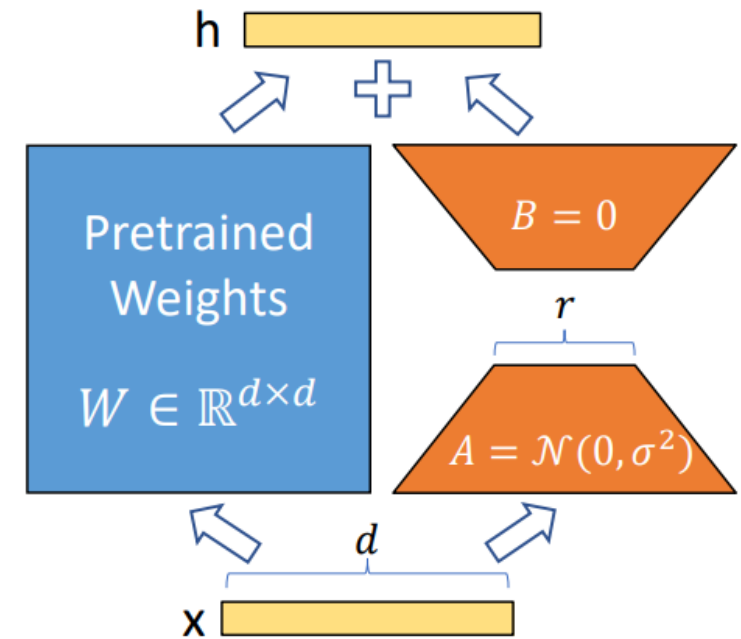
Task Adaption – Supervised Fine-Tuning

- ▶ Modern LLMs (e.g., Llama models) can have up to billions of parameters.
 - ▶ They often have hundreds or even thousands of layers of multi-head attention and feed-forward layers.
- ▶ It's not realistic to fine-tune all of them.
- ▶ We need a **parameter-efficient fine-tuning (PEFT)** method.



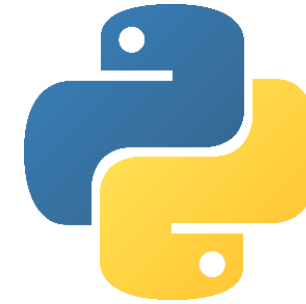
Task Adaption – Supervised Fine-Tuning

- ▶ Low-Rank Adaptation (LoRA) is a common PEFT method (Hu et al. 2021).
- ▶ Instead of updating W to W' directly, LoRA only updates the parameter changes ΔW only, where:
$$W' = W + \Delta W$$
- ▶ ΔW can be decompose into two matrices A and B , with an intrinsic *rank* r capturing the “essence” of parameter updates.
 - ▶ r is much smaller than the original dimension, this significantly reduce the number of parameters being updated.



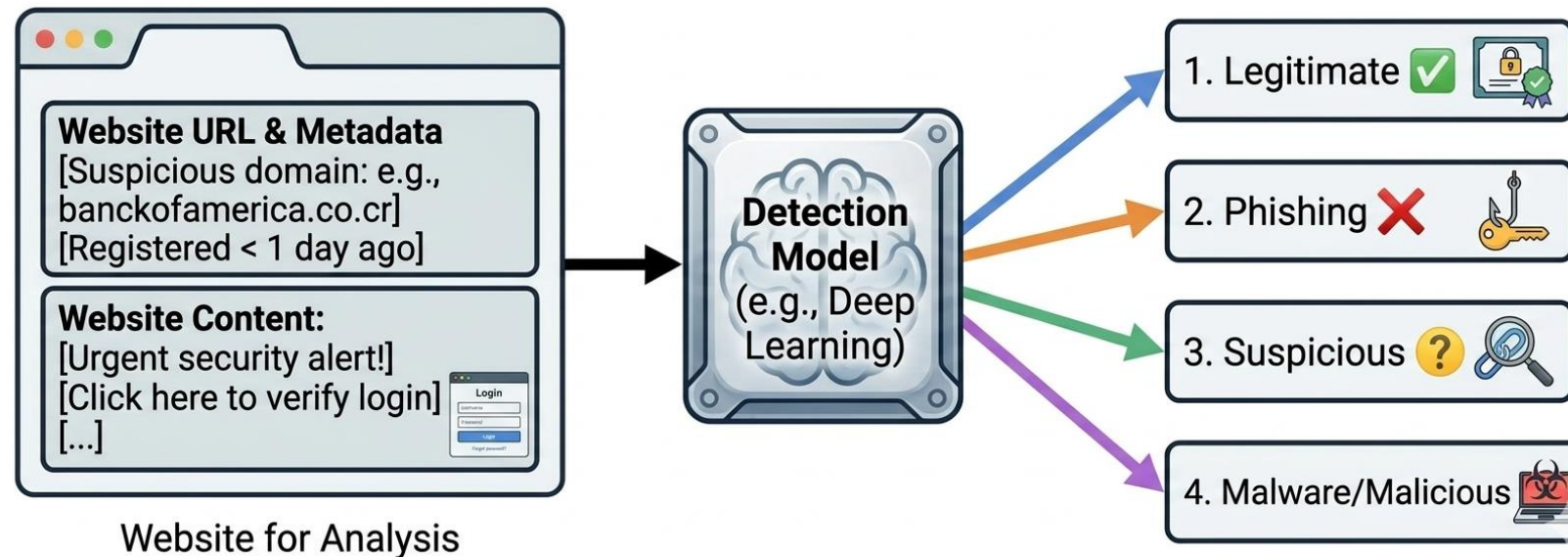
LoRA Reparametrization
(from Hu et al. 2022)

Python Example



Hugging Face

- ▶ **Python example:** Phishing Detection (revisited) with Transformer models
- ▶ **Data:** Prior phishing websites, focusing on the html source code
- ▶ **Key Packages:** PyTorch, transformer



Online M.S. in AI



Review Questions

- ▶ In a binary classification task, what activation functions should be adopted in the hidden layer and output layer, respectively?
- ▶ How does CNN efficiently learn topological patterns?
- ▶ How does the attention mechanism capture “context?”
- ▶ What is the main difference between the standard Transformer model and GPT that allows the latter to exhibit emergent reasoning capabilities?
 - ▶ Can you think of why this is the case?



Next Week

